Universidade de Aveiro Departamento de Engenharia Electrónica e Telecomunicações
2005

Sérgio Pedro Ferreira Pinheiro

Impacto da Rede de Comunicação em Sistemas de Controlo Distribuídos


Universidade de Aveiro
2005
Departamento de Engenharia Electrónica e Telecomunicações

Sérgio Pedro Ferreira Pinheiro

Impacto da Rede de Comunicação em Sistemas de Controlo Distribuídos

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. Luís Miguel Pinho de Almeida, Professor Auxiliar do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro.

Dedico este trabalho à minha família pelo incansável apoio.

o júri

presidente

Prof. Doutor José Alberto Gouveia Fonseca
Professor Associado da Universidade de Aveiro

Prof. Doutor Carlos Baptista Cardeira
Professor Auxiliar do Instituto Superior Técnico / Universidade Técnica de Lisboa

Prof. Doutor Alexandre Manuel Moutela Nunes Mota
Professor Associado da Universidade de Aveiro

Prof. Doutor Luís Miguel Pinho de Almeida
Professor Auxiliar da Universidade de Aveiro

agradecimentos

Ao Professor Doutor Luís Miguel Pinho de Almeida, queria agradecer a sua paciência, orientação, disponibilidade constantes.

Aos meus pais, pelo incansável apoio e compreensão que sempre me deram em todos os momentos.

À minha namorada Sara, a paciência e apoio nos momentos mais difíceis e de impaciência.

Aos meus amigos pelo apoio e conselhos úteis que me deram.

À Escola Superior de Tecnologia, do Instituto Politécnico de Castelo Branco, a dispensa de serviço por um semestre.

palavras-chave

Sistemas de tempo-real; Malha de Controlo; Arquitectura Distribuída; Arquitectura Centralizada; Sistemas de Controlo Distribuídos; Sistemas de Controlo Embutidos Distribuídos; Barramento; Disparo por Eventos; Disparo por Tempo; Amostras Omissas; Período de Controlo; Erros Transitórios; *jitter*; Atraso no Controlo; *Token-Passing*; *Ethernet*; *Deadlines*; *Multirate*; *Kernel*; *Time Slots*; *Link Active Scheduler*; *Link Masters*.

resumo

O conceito de controlo através de uma rede de comunicação tem aplicações vastas. Em muitos casos, os sensores, controladores e actuadores estão fisicamente distribuídos, sendo necessário que troquem entre si a informação de controlo utilizando uma rede de comunicação, por forma a que o sistema execute uma determinada lei de controlo sobre um determinado processo (e.g., controlo da velocidade de um motor, controlo da injecção de combustível num motor de um automóvel, controlo da posição do braço de um robot, etc.)

Entretanto, quando se usa uma rede partilhada para transmitir a informação de controlo, o atraso entre o tempo de aquisição de um sinal (amostragem) e o instante de aplicação do respectivo sinal de controlo (actuação) é influenciado por múltiplos parâmetros, desde a velocidade de transmissão da rede (*bitrate*), ao comprimento da mensagem, e também à quantidade de tráfego na rede e ao protocolo usado no acesso ao meio de comunicação. Assim, num sistema de controlo distribuído, é inevitável o aparecimento de uma latência elevada, quando comparada com a correspondente latência num sistema não distribuído, a qual é induzida pela rede e poderá causar degradação do desempenho e estabilidade do controlo.

O objectivo desta dissertação é analisar esse efeito de latência induzido pela rede, sob a influência de vários padrões de tráfego e de vários protocolos de acesso ao meio (*MACs*), usando para tal a ferramenta de simulação *TrueTime*, que facilita a simulação de modelos simples de sistemas distribuídos de controlo.

keywords

Real Time Systems; Control Loop; Distributed Architecture; Centralized Architecture; Distributed Control Systems; Embedded Distributed Control Systems; Fieldbus; Event-Triggered; Time-Triggered; Vacant Samples; Control Period; Transient Errors; Jitter; Control Delay; Token-Passing; Ethernet; Deadlines; Multirate; Kernel; Time Slots; Link Active Scheduler; Link Masters.

abstract

The concept of networked control has many applications. For example, when sensors, controllers and actuators are physically distant from each other, the control information needs to be exchanged over a communication network so that the system achieves a certain control objective (e.g., engine speed control, engine fuel injection control, robot arm position control).

When a network is used to transmit control data, the delay between the sampling time and actuation time includes the transmission of several messages and is determined not only by the speed of the network (bitrate) and message size, but also by the amount of other traffic on the network and the medium access control protocol used. So, in a distributed control system it is inevitable to observe a higher control latency when compared with the latency in a non distributed system, which is induced by the network and can cause the degradation of the control performance and even loss of stability.

The main goal of this dissertation is to analyze the effect induced by the network latency due to the influence of network load and medium access control protocols (MACs). To do such work we use the simulation tool TrueTime that facilitates the simulation of simple distributed control loops and the assessment of the respective control performance.

ÍNDICE

Capítulo I

Introdução

1. Resumo

A era da *computação moderna, com início sensivelmente na* década de 40, veio permitir utilizar em diversas áreas os dispositivos de cálculo que entretanto se começaram a desenvolver, vulgo computadores. Foi o caso, por exemplo, dos sistemas de controlo automático em que um computador executa em ciclo infinito um algorítmo de controlo sobre um fluxo de dados sensoriais, gerando um fluxo de actuações a aplicar ao sistema a controlar. Estes sistemas de controlo automático computorizado, até cerca de meados da década de 70, eram todos baseados em Arquitecturas Centralizadas, i.e., usavam apenas um computador com uma única unidade processadora à qual se ligavam todos os sensores e actuadores do sistema, bem como consolas de operação, painéis de sinalização, impressoras, etc.. Estas arquitecturas, contudo, apresentam diversos problemas como o uso de imensa cablagem, pois todos os dispositivos que se encontram dispersos pelo ambiente a controlar estão ligados à unidade central de processamento através de cabos, grande

sensibilidade ao ruído uma vez que a maior parte das comunicações era analógica, o comprometimento de funcionamento de todo o sistema no caso de falha do computador e restrições de expansão associadas à capacidade física de interconexão da unidade central (Fonseca, 1999)

A partir de meados da década de 70 surgem avanços tecnológicos fundamentais como o desenvolvimento de microprocessadores poderosos e interfaces de comunicação série de baixo custo. Estes factores, associados quer a uma pressão económica quer a legislação laboral e ambiental cada vez mais restritivas, conduzem a indústria a aumentar os seus níveis de automatização numa perspectiva de reduzir custos e melhorar o controlo de qualidade de produtos e processos. Os problemas inerentes a arquitecturas centralizadas tornaram-se evidentes e começaram a aparecer outras soluções arquitecturais baseadas na distribuição das funcionalidades do sistema por várias unidades processadoras interligadas entre si por uma rede de comunicação, originando os *Sistemas de Controlo Distribuídos*.

Neste tipo de arquitecturas, o poder computacional encontra-se potencialmente distribuído quer pelos vários sensores e actuadores, quer pelas várias funções de controlo, cálculo e supervisão que se encontram distribuídas pelos vários elementos do sistema, os quais são designados por nós ou nodos.

Com o aumento da integração electrónica, torna-se possível construir nós de dimensões bastante reduzidas com interface I/O e um ou mais interfaces de comunicação. Assim, as reduzidas dimensões destes nós, permitem-lhes serem colocados dentro dos próprios sistemas a controlar, mesmo quando estes também são de dimensões reduzidas. Este processo de implementar um sistema de controlo no interior do próprio sistema a controlar, munido de uma interface de operação que suporta apenas a funcionalidade restrita que se pretende executar, deu origem aos *Sistemas de Controlo Distribuídos* e *Embutidos* (*Distributed Embedded Control Systems*). Neste tipo de sistemas a implementação distribuída surge em aplicações que variam desde o controlo de um automóvel, ao de um avião, de máquinas ferramentas, de robôs, de instrumentação médica, ou até ao controlo de experiências em laboratórios.

De uma forma geral um sistema de controlo distribuído e embutido possui uma ou mais malhas de controlo realimentado (implementadas através de um conjunto de dispositivos: sensores, actuadores, controladores, etc.) fechadas através de uma rede partilhada (tipicamente designada *fieldbus* em controlo de processo). Sendo a malha de controlo implementada de uma forma distribuída, com os diversos dispositivos a trocar informação através da rede, irão surgir atrasos associados à comunicação da informação de controlo, quer devidos à transmissão dos dados quer a erros e

omissões. Estes atrasos são ainda influenciados pelo estado de sobrecarga da rede e pelo protocolo de comunicação utilizado, podendo causar a degradação do desempenho da qualidade do controlo do sistema e no limite, a respectiva instabilidade. Desta forma, um sistema de controlo distribuído só é eficaz se estes atrasos temporais estiverem adequadamente limitados. Estes requisitos fazem com que os sistemas de controlo sejam inerentemente *Sistemas de Tempo-Real* uma vez que poderão resultar graves consequências se não forem satisfeitas as suas necessidades de precisão temporal, quer nos cálculos quer nas comunicações.

Tipicamente um sistema de tempo-real consiste num *sistema que controla* (sistema de controlo) e num *sistema controlado* (processo). O sistema de controlo interage com o processo baseado na informação recolhida a partir de vários sensores a ele ligados. É assim imperativo que o estado do processo, visto pelo sistema de controlo, seja consistente com o seu estado actual, i.e., que desde o instante em que o processo é amostrado pelo(s) sensor(es) até à recepção desses dados pelo controlador, respectivo processamento e envio dos sinais de controlo para o(s) actuador(es), não passe demasiado tempo relativamente ao previsto no projecto do controlador. Caso contrário, os efeitos das actuações do sistema de controlo poderão ser desastrosas.

A necessidade de precisão temporal em sistemas de tempo-real surge também devido a outro tipo de impacto físico das acções de controlo sobre o processo, para além das decorrentes da instabilidade de malhas de controlo realimentado. Por exemplo, se o sistema que controla um robô não lhe der a ordem a tempo para parar ou virar numa direcção específica, o robô poderá colidir como outro objecto no ambiente de operação. Desnecessário será dizer que tal infortúnio pode resultar numa catástrofe de grandes dimensões.

A obtenção da desejada precisão temporal é dificultada quando se usa uma rede partilhada para transmitir informação de controlo. Isto é, factores como a velocidade de transmissão (*bitrate*), tamanho da mensagem, protocolo de controlo de acesso ao meio e quantidade de tráfego na rede são condicionantes de grande importância no atraso temporal entre o instante em que é amostrado um processo e o instante de aplicação do sinal de controlo (actuação) nesse mesmo processo. Assim, num sistema de controlo distribuído, é inevitável a existência de uma elevada latência de controlo quando comparada com a latência correspondente num sistema não distribuído, a qual está essencialmente relacionada com os atrasos de comunicação.

2. APRESENTAÇÃO DO PROBLEMA

Na sequência do exposto na secção precedente, ficou claro que a concretização eficaz de sistemas de controlo distribuídos requer uma cuidada análise da latência de controlo a qual, por sua vez, é substancialmente influenciada por aspectos relacionados com a comunicação.

Assim, pretende-se nesta dissertação estudar o impacto da comunicação no desempenho de malhas distribuídas de controlo realimentado. Em particular, focar-se-á nas redes de comunicação baseadas em barramento eléctrico série com meio partilhado, e nos aspectos da arquitectura funcional (atribuição a nós físicos das funções de sensor, controlador e actuador), nos paradigmas de disparo das comunicações (por eventos e por tempo) e nos protocolos de acesso ao meio (MAC – Medium Access Control).

Para analisar o impacto da rede de comunicação na estabilidade de um sistema de controlo distribuído, são úteis ferramentas de software para simular o funcionamento do sistema e mensurar o desempenho do controlo, tendo em conta os atrasos e *jitter* que afectam o fluxo de informação.

O *TrueTime* é uma dessas ferramentas que facilita a simulação do comportamento temporal de um *kernel* na execução de tarefas de controlo, e ainda de modelos simples de redes de comunicação e sua influência em malhas de controlo distribuído.

Serão criados vários cenários de forma a analisar os atrasos devido à rede de comunicação sobre as diferentes configurações dos dispositivos que constituem o sistema de controlo.

O primeiro cenário terá como objectivo o estudo do comportamento da malha de controlo, inicialmente sem o uso de rede de comunicação, i.e., modelo Sensor + Controlador + Actuador num único nó e, posteriormente, com a rede a interligar os vários nós, começando pelo modelo com dois nós em que o actuador se situa num nó e o sensor e controlador noutro, e seguidamente pelo modelo totalmente distribuído, com três nós, um para cada funcionalidade. Em ambas as simulações não existirá carga adicional e será usado o protocolo de acesso ao meio *CSMA*/*AMP* (utilizado na *Controller Area Network – CAN*) a uma taxa de transmissão de 125 kbps.

O segundo cenário de simulação consistirá no estudo do modelo em que os dispositivos sensor, actuador e controlador, se encontram dispersos pela rede (modelo totalmente distribuído com três nós). Numa primeira experiência será adicionado um

nó que gerará tráfego de interferência com alta prioridade mas carga relativamente baixa. Numa segunda experiência, será aumentado o nível de interferência induzida na rede por esse nó aumentando a carga de tráfego submetido. Outra experiência consistirá em adicionar mais um nó à rede, com prioridade inferior à dos dispositivos sensor e controlador. Uma última experiência terá como objectivo colocar ambos os nós de interferência com prioridades superiores às dos nós da malha de controlo e verificar o comportamento da malha de controlo nessas circunstâncias.

No último cenário serão realizadas as mesmas experiências que foram executadas para os cenários anteriores, no entanto para diferentes protocolos de acesso ao meio, que serão: *CSMA*/*CD* (semelhante ao de *Ethernet*); *Round Robin* (semelhante ao *Token Bus*); e *TDMA* (como no *TTP- Time Triggered Protocol*). Por fim será feita a análise dos resultados obtidos por simulação para os diferentes cenários.

3. ESTRUTURA DA DISSERTAÇÃO

Nesta secção apresentamos a estrutura da Dissertação, relativamente aos restantes capítulos. No capítulo II, começaremos por apresentar as características gerais de dois tipos de arquitecturas usadas em sistemas de controlo, i.e., centralizada e distribuída. Será ainda referido que com o aumento dos níveis de integração electrónica é possível construir nós com muito poucos componentes, onde as suas reduzidas dimensões e custos permitem instalá-los de forma distribuída, perto da origem ou destino dos sinais analógicos, i.e., integrados nos sensores e actuadores, e dentro de máquinas ou outros sistemas a controlar, mesmo de relativamente pequenas dimensões, originando os sistemas de controlo distribuídos e embutidos. Serão abordadas ainda várias possibilidades de mapear as diversas funcionalidades de um sistema de controlo distribuído (sensor, controlador e actuador) na arquitectura de hardware uma vez que cada mapeamento possui requisitos computacionais e de comunicação diferentes, resultando em diferentes comportamentos temporais e diferentes desempenhos da malha de controlo.

O capítulo III irá tratar de duas filosofias ou *paradigmas* fundamentais relacionados com o modo como as acções e transacções num sistema são disparadas. Esses dois paradigmas têm um impacto profundo nas propriedades do sistema global e são designados por disparo por eventos (*event-triggered*) e disparo por tempo (*time-triggered*). Serão também referidas as principais características de um paradigma relativamente ao outro. Por fim será feita uma análise de ambos os paradigmas na perspectiva dos sistemas de controlo distribuído.

Quando o canal de comunicação é partilhado, é necessário resolver o problema de determinar quem pode transmitir em cada instante, para que não haja colisões e perda de informação. Este problema é genericamente conhecido como o controlo de acesso ao meio, existindo muitos protocolos para o resolver, alguns dos quais serão descritos no capítulo IV.

No capítulo V serão realizadas análises comparativas baseadas em simulação de sistemas reais. As simulações serão efectuadas com recurso ao software *TrueTime* que funciona no ambiente *MATLAB*/*Simulink* e que facilita a simulação de todo o sistema, tendo em conta o comportamento temporal quer de um *kernel* multitarefa de tempo-real em cada nó, quer de uma rede de comunicação a interligar os nós, permitindo analisar a influência dos atrasos respectivos em malhas distribuídas de controlo. As conclusões desta dissertação serão apresentadas no capítulo VI.

Capítulo II

Arquitecturas dos Sistemas de Controlo

1. De Arquitecturas Centralizadas a Distribuídas

Desde os anos 70 que uma pressão económica crescente aliada a legislação laboral e ambiental cada vez mais restritiva, vêm obrigando as indústrias a aumentar os seus níveis de automatização de forma a reduzir custos e melhorar o controlo de qualidade de produtos e processos.

Embora o uso de equipamento automatizado na indústria tenha já muitos anos, só mais recentemente se tem generalizado devido ao baixo custo dos dispositivos semicondutores que com capacidade de programação e comunicação lhes permitem ser integrados e aplicados em larga escala nos processos de automatização industrial e, em particular, de controlo automático. Com este nível de integração é possível recolher informação mais completa, correcta e actual sobre o estado do processo, controlá-lo de forma mais eficaz quer em termos da qualidade do controlo,

isto é, menor erro, quer em termos de outras variáveis como o consumo de energia. Por outro lado, a integração e aplicação em larga escala requer uma arquitectura adequada que sustente as interacções necessárias à concretização dos objectivos globais do sistema. A arquitectura dos sistemas de automatização industrial e de controlo tem vindo a evoluir ao longo dos tempos, havendo hoje um claro domínio das arquitecturas distribuídas, as quais apresentam inúmeros benefícios relativamente a arquitecturas mais antigas, baseadas num modelo centralizado de controlo e operação. Neste capítulo abordaremos estas arquitecturas por ordem cronológica, começando pelas centralizadas e posteriormente as distribuídas, discutindo os respectivos benefícios. Abordaremos depois os sistemas de controlo distribuídos na perspectiva de tempo-real. Seguidamente faremos uma caracterização dos requisitos temporais numa aplicação de controlo distribuído. Por fim serão referidos vários modelos de um sistema de controlo distribuído assinalando os constrangimentos temporais associados a cada um deles.

1.1. Arquitectura Centralizada

Até meados dos anos 80, a arquitectura tipicamente usada nos sistemas de automatização industrial e de controlo era centralizada. Esta arquitectura é caracterizada pela existência de um ponto central para onde convergem todos os sinais recolhidos do processo, onde é feito todo o processamento de controlo e onde são gerados todos os comandos a ser aplicados ao processo. Assim, estes sistemas exibem uma topologia em estrela, conforme mostrado na figura 2-1 (Almeida *et al.*,1998).

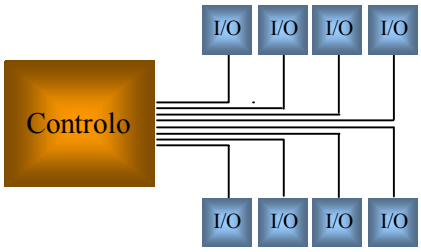

Figura 2-1 – Arquitectura centralizada

Se bem que simples, pois toda a informação está centralizada e é coordenada sincronamente por uma só unidade processadora, tornando mais simples o controlo do sistema, esta arquitectura apresenta alguns problemas, havendo situações em que não é conveniente, por exemplo, quando a dispersão de sensores e actuadores, i.e. dispositivos de I/O, é significativa. Nesta situação, a necessidade de fazer passar um cabo do controlador a cada dispositivo de I/O, traz vários problemas consigo:

- Uma grande concentração de cablagem junto do controlador, com os problemas de atravancamento inerentes, resultando em dificuldade de instalação e manutenção;

- A dimensão do painel de interligação, normalmente um bastidor, de elevado custo e de fiabilidade naturalmente condicionada pela relação dimensão/número de contactos;

- Uma extensão potencialmente muito grande da cablagem total, com um elevado custo associado;

- Sensibilidade ao ruído, uma vez que uma parte significativa das transmissões de I/O era analógica.

Outro problema das arquitecturas centralizadas é a baixa fiabilidade inerente à existência de um ponto singular de falha, i.e. o controlador. A sua avaria compromete o funcionamento de todo o conjunto. Este aspecto é particularmente crítico quando, pela sua natureza ou condicionantes económicas, a interrupção do processo por avaria de um componente não seja aceitável, e.g. sistemas de segurança crítica (*safety-critical systems*) tais como sistemas de segurança em centrais atómicas ou indústrias químicas perigosas, geração de energia em hospitais, controlo de tráfego aéreo, etc (Kopetz, 1997).

Outro aspecto negativo das arquitecturas centralizadas é a limitação à respectiva capacidade de expansão. O número de entradas e saídas do controlador pode estar limitado. Mais entradas e saídas significam mais processamento, aumentando o tempo de ciclo, podendo mesmo dar-se o caso da unidade central de processamento não ter capacidade de tratar o novo programa (quer devido à sua extensão, quer devido à sua complexidade) (Kopetz, 1997).

1.2. ARQUITECTURA DISTRIBUÍDA

A solução para os problemas apontados na secção anterior passa necessariamente pela adopção de outras soluções arquitecturais, tal como aquelas que fazem uso de distribuição originando os chamados *Sistemas de Controlo Distribuídos*. Nestas arquitecturas, que passaram a ser usadas a partir de meados dos anos 80, o poder computacional está distribuído pelos vários sensores e actuadores, os quais estão também dotados de capacidades comunicantes através de sistemas de comunicação digital com partilha de canal. Por sua vez, também as várias funções de controlo, cálculo e supervisão são distribuídas pelos vários elementos do sistema, normalmente designados nós ou nodos.

A figura 2-2 procura pôr em evidência o conceito de distribuição do algoritmo de controlo. A figura da esquerda representa um sistema centralizado. Mesmo havendo distribuição dos componentes, pois os sensores, actuadores e interfaces com o operador podem estar distantes da unidade central, o seu sinal é trazido até esta, onde é feito todo o processamento.

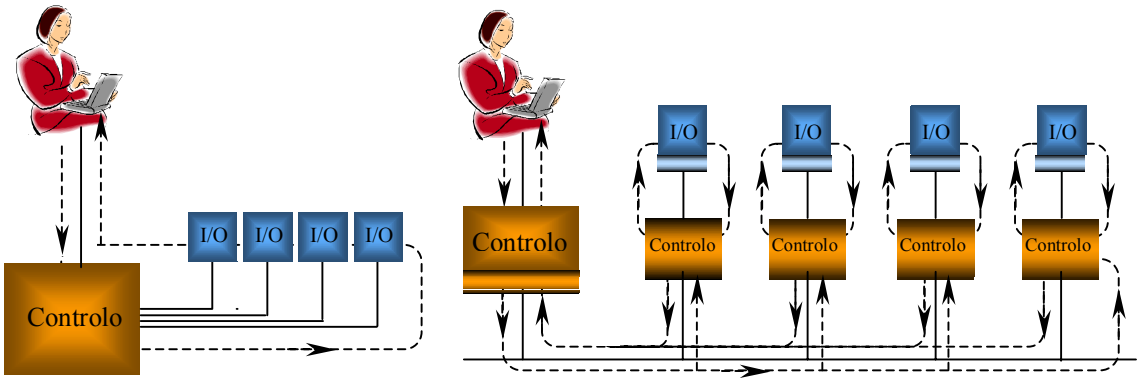

Figura 2-2 – Distribuição do algoritmo de controlo

Na figura da direita, cada elemento periférico (sensor, actuador ou interface com o operador) tem inteligência própria no sentido de que dispõe de capacidade de processamento autónoma. Cada um destes elementos (nós) faz a interface com o processo através dos respectivos componentes de I/O e com os restantes nós do sistema através de um sistema de comunicação (rede). Este sistema de comunicação suporta todas as trocas de informação entre os vários nós, conforme exigido pela aplicação global, e é frequentemente designado por barramento de campo (*fieldbus*) dado interligar os dispositivos de campo (sensores e actuadores).

Em termos conceptuais, os aspectos mais contrastantes com a arquitectura centralizada são o da utilização frequente de um sistema de comunicação partilhado, o barramento de campo, que permite introduzir uma grande simplificação da cablagem e redução do respectivo custo, e também o da distribuição do algoritmo de controlo, o qual pode ser dividido em várias partes que executam paralelamente em cada um dos nós (Mota *et al*., 1999).

O algoritmo de controlo já não se executa apenas numa única unidade controladora, onde é relativamente fácil sincronizar todas as actividades a desempenhar, mas passou a ser composto por múltiplos programas que requerem sincronização e comunicação adequadas. Se por um lado as arquitecturas distribuídas trouxeram uma simplificação da cablagem bem como outras propriedades desejáveis referidas mais adiante, também é certo que trouxeram um grau de complexidade bastante superior,

que tem de ser adequadamente gerido para permitir obter reais benefícios (Almeida *et al*., 1998).

1.2.1. ARQUITECTURAS DISTRIBUÍDAS EMBUTIDAS

Ao longo dos anos 90, com o aumento dos níveis de integração electrónica tornou-se possível construir nós com muito poucos componentes, em torno de um microcontrolador, com dimensões reduzidas mas dispondo de um interface de I/O e

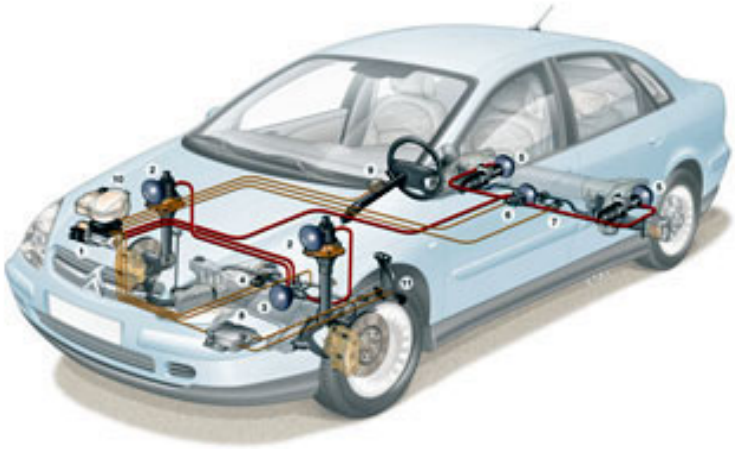

Figura 2-3 – Sistemas embutidos distribuídos. Sistema ABS.

de um, ou mais, interfaces de comunicação. As reduzidas dimensões e custos destes nós permitiram multiplicá-los e instalar arquitecturas distribuídas de controlo mesmo dentro de máquinas ou outros sistemas a controlar, figura 2-3, originando o que se costuma designar por sistemas embutidos distribuídos (*distributed embedded systems*). O controlo do sistema está assim entregue a um conjunto de múltiplos nós, todos com capacidade de cálculo, embutidos no próprio sistema a controlar, e em que a rede de comunicação é como um sistema nervoso que percorre todo o sistema e onde circula toda a informação. Deste modo, integram-se o subsistema a ser controlado com o subsistema de controlo, diminuindo a distância entre os sensores e actuadores e os pontos onde é executado o algoritmo de controlo (Almeida *et al*., 1998).

1.2.2. BENEFÍCIOS DAS ARQUITECTURAS DISTRIBUÍDAS

As principais vantagens que poderão advir da utilização de arquitecturas distribuídas são:

- Filtragem da informação. Dado que os nós possuem capacidade de cálculo, podem ser utilizados para tratar a informação recolhida do processo logo na fonte, linearizando-a e digitalizando-a, tornando-a menos susceptível a erros durante a subsequente transmissão. Por outro lado, a capacidade de cálculo dos nós permite também diminuir a quantidade de informação transportada pela rede. Por exemplo, ao fechar localmente (no nó) uma malha de controlo, passa a ser necessário unicamente receber o *set-point* e responder a pedidos de informação sobre o estado actual (Almeida *et al*., 1998).

- Modularidade. A capacidade de processamento dos nós também permite que lhes seja atribuída a execução de funcionalidades específicas associando um módulo de controlo a uma parte específica do processo. Este nível de modularidade em que o sistema é composto por vários nós fisicamente independentes, que facilmente podem ser trocados, facilita grandemente a manutenção e evolução destes sistemas. No limite, desde que as interfaces se mantenham, essas alterações não implicam qualquer revisão geral do projecto do sistema (Almeida *et al*., 1998).

- Manutenção da cablagem. Quando se usa um sistema de comunicação partilhado a cablagem fica muito simplificada trazendo imensas vantagens. Por exemplo, num sistema baseado em barramento de campo, é possível fazer circular um número elevado de sinais diferentes sobre o mesmo meio, geralmente um cabo com dois ou quatro condutores. A instalação e substituição da cablagem é também muito fácil e rápida.

- Tolerância a falhas. Os sistemas distribuídos introduzem a capacidade de replicar o mesmo programa em vários nós criando redundância espacial. Deste modo, se uma das réplicas falhar uma outra pode tomar o seu lugar e substituí-la, até à reparação do componente avariado, mantendo o sistema em funcionamento.

- Escalabilidade. Se for necessário aumentar o número de pontos de I/O ou mesmo introduzir novas funcionalidades, é normalmente possível acrescentar mais nós, eventualmente segmentando a rede e recorrendo a *bridges* ou *gateways*.

2. A PERSPECTIVA DOS SISTEMAS DISTRIBUÍDOS DE TEMPO-REAL

Tipicamente, um sistema de controlo distribuído embutido tem uma ou mais malhas de controlo fechadas através de uma rede partilhada (*fieldbus*), como pode ser visto na figura 2-4.

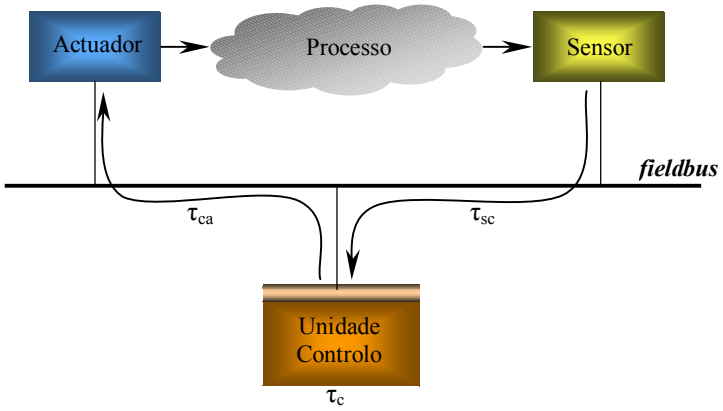

Figura 2-4 – Modelo básico de uma malha fechada de controlo. τ_{sc}, τ_{ca} e τ_c representam respectivamente os atrasos na comunicação sensor-controlador e controla- dor-actuador, e o tempo de execução do controlo

Cada malha é implementada através de um conjunto de dispositivos: sensores, actuadores, controladores, etc., que trocam informação através dessa rede. Nestes sistemas a rede partilhada tem uma influência determinante no desempenho das malhas de controlo já que quaisquer atrasos ou erros introduzidos pelo sistema de comunicação afectam directamente a pontualidade da informação de controlo.

As fases principais de operação de uma malha de controlo são a amostragem efectuada pelo(s) sensor(es), o controlo efectuado pelo controlador, e a actuação efectuada pelo(s) actuador(es). A amostragem deve ser realizada no mesmo instante relativo em cada período de amostragem, o controlo deve-se iniciar e finalizar rapidamente após a amostra estar disponível e a actuação deve ocorrer imediatamente após o controlo, ou num instante fixado após ter sido realizada a amostragem (dependendo de como foi projectado o controlador). Em sistemas monoprocessador, é comum assumir-se que a passagem de uma fase à seguinte é instantânea, i.e. sem tempos suplementares de espera entre elas. Por vezes também

se despreza o tempo de execução do sensor e do actuador, que normalmente executam operações simples (Martí *et al*., 2000).

Entretanto, sendo a malha de controlo implementada de uma forma distribuída, com os vários dispositivos a trocar informação através da rede, tal suposição poderá não se verificar devido aos tempos necessários para a transmissão da informação, figura 2-4. Estes atrasos suplementares poderão causar a degradação da qualidade do controlo, podendo o sistema, no limite, tornar-se instável (Martí *et al.*, 2000). Assim, um sistema de controlo distribuído só é eficaz se os atrasos entre amostragem e geração do sinal de controlo (τ_{sc}) e entre este e a actuação (τ_{ca}) estiverem adequadamente limitados.

Estas limitações temporais relativas ao tempo total entre amostragem e actuação fazem com que os sistemas de controlo sejam inerentemente *Sistemas de Tempo-Real*. Nestes sistemas a validade de um resultado não depende só da correcção lógica, mas também do instante em que o resultado é produzido. Isto é, deverão ser capazes de interagir com o ambiente (processo) de uma forma temporalmente ajustada à dinâmica deste e de forma previsível (*predictable*). Uma vez que os problemas temporais condicionam a qualidade do controlo, será feita uma caracterização dos respectivos parâmetros na secção seguinte. Por fim serão referidas várias arquitecturas operacionais para um sistema de controlo distribuído evidenciando os constrangimentos temporais associados a cada uma delas.

2.1. CARACTERIZAÇÃO DOS PROBLEMAS TEMPORAIS

No desenvolvimento de um sistema distribuído de tempo-real, um dos principais objectivos é encontrar uma implementação que satisfaça os requisitos temporais da aplicação (Sanfridson, 2000). Estes requisitos representam limites máximos que os respectivos parâmetros temporais em operação não deverão ultrapassar e podem ser vistos como perturbações máximas que influenciam o sistema. Desta forma, o desempenho, controlabilidade, robustez, observabilidade e estabilidade de uma malha de controlo dependem, além de outros factores, desses requisitos temporais (Sanfridson, 2000). Tipicamente são considerados, o atraso de controlo (*control delay*) também referido como atraso amostragem-actuação, a variação deste atraso (*control jitter*), erros transitórios (*transient errors*) e o período de controlo (*control period*).

2.1.1. ATRASO DE CONTROLO (*CONTROL DELAY*)

Control Delay é o atraso entre o instante de amostragem e o correspondente instante de actuação. É também referido como sendo o atraso amostragem actuação (*sampling-actuation delay*) (Sanfridson, 2000).

Os atrasos no controlo surgem da execução do algoritmo de controlo, conversões A/D e D/A e da rede de comunicação. O atraso induzido quer pelas conversões A/D e D/A, quer pela execução do algoritmo de controlo é previsível, sendo muitas vezes considerado constante e adicionado no projecto e análise da malha de controlo. Se o atraso induzido pela rede entre o sensor-controlador, τ_{sc}, e entre o controlador-actuador, τ_{ca}, for constante, também pode ser adicionado. Para lidar com esses atrasos, que muitas vezes não são constantes por razões operacionais, como variações no tempo de execução dos algoritmos, a execução de outras tarefas nos nós ou a presença de tráfego adicional na rede, é comum utilizar-se um majorante absoluto, muitas vezes igual a um período de controlo. No entanto, do ponto de vista do controlo, o desempenho e a estabilidade são adversamente afectados quando é usado um grande atraso mesmo que constante, i.e., se o atraso for grande, mesmo que o sistema seja estável perde em desempenho, e.g. rapidez de convergência em mudanças de *set-point* (Sanfridson, 2000). Portanto não deve ser introduzido atraso desnecessário quando se implementa uma lei de controlo. Como regra prática, o instante de actuação deve surgir, logo que possível, após o de amostragem, no entanto existem algumas razões para inserir um atraso fixo entre amostragem e actuação:

1) Diminuição da quantidade de *control jitter* já que o atraso fixo permite absorver variações nos atrasos parciais que constituem o *control delay*;

2) Minimização da quantidade de amostras perdidas (*vacant samples*), que surgem devido a erros na rede ou dessincronização entre transmissor e receptor (e.g. quando o receptor se adianta em relação ao transmissor);

3) Simplificação do escalonamento das várias tarefas envolvidas na malha de controlo, permitindo, por exemplo, utilizar a facilidade de definição de *off-sets* típica em muitos sistemas operativos de tempo-real.

No entanto, o problema reside em escolher este atraso fixo de forma a optimizar o desempenho do sistema sem colocar em risco a sua estabilidade.

2.1.2. PERÍODO DE AMOSTRAGEM (*CONTROL PERIOD*)

O período de controlo (*Control Period*) ou o período de amostragem é um parâmetro fundamental na implementação de um sistema de controlo, não devendo nem ser muito curto nem muito longo. Um período de amostragem curto, em geral, aumenta o desempenho do controlo (Sanfridson, 2000) já que melhora a qualidade da representação digital dos sinais analógicos amostrados. Isto é verdade particularmente no caso dos controladores projectados de forma analógica e posteriormente

digitalizados. No entanto, um período curto também aumenta a taxa de utilização dos recursos do sistema para a execução do algoritmo de controlo, podendo aumentar a interferência mútua com outras actividades, provocando aumentos de tempo de resposta e de *jitter*. Por outro lado, se o período de amostragem for longo, maior será o tempo (entre amostras) em que a informação do estado do sistema não é actualizada, o que poderá permitir a propagação de perturbações transitórias e aumentar o erro do controlo (MacGregor, 1976). Se for longo demais porá a controlabilidade do sistema em risco. Em geral, a escolha do período de controlo deverá ser baseada na dinâmica do respectivo processo mas escolhendo o maior valor que permite obter o desempenho pretendido (MacGregor, 1976).

Também é possível realizar uma amostragem não-uniforme. Esta é baseada na ideia de que a necessidade de controlo num sistema realimentado não é constante. Haverá vezes em que o sistema está estável e em que necessita apenas de intervenções esporádicas do sistema de controlo e outras em que o sistema passa por estados de grande dinâmica e erro, requerendo um controlo apertado (com elevada cadência) para manter a respectiva saída dentro dos valores desejados (Miśkowicz, 2001). A principal razão de usar uma amostragem não-uniforme é a de reduzir a utilização total dos recursos do sistema atribuindo períodos de amostragem de acordo com a dinâmica instantânea do processo. Este tipo de amostragem pode ser realizado de duas formas, usando intervalos variáveis em que o controlo e actuação são disparados por eventos (alterações significativas do estado do sistema) (Miśkowicz, 2001) ou usando múltiplos intervalos relacionados com múltiplas malhas de controlo encadeadas, designada por *multirate* (Magalhães *et al*., 2002). No entanto a teoria sobre amostragem disparada por eventos encontra-se ainda em desenvolvimento, ao contrário da *multirate*, já bastante estudada (Sanfridson, 2000). A abordagem *mutirate* é muito comum em sistemas distribuídos, em que alguns nós executam diferentes controladores a ritmos diferentes (e.g. controlo de postura e controlo de velocidade de cada uma das rodas num robô (Magalhães *et al*., 2002)). Por outro lado, em (Martí *et al,* 2002) encontramos um exemplo recente de utilização de amostragem não-uniforme mas utilizando um conjunto finito de valores para o período. Neste caso, o objectivo é acomodar as variações no período de amostragem provocadas pelo escalonamento e execução concorrente de outras tarefas no mesmo processador, e comutar dinamicamente os parâmetros do controlo de acordo com o período instantâneo de amostragem.

2.1.3. *JITTER*

O *jitter*, variação dos atrasos e do período de controlo, poderá afectar negativamente a estabilidade e o desempenho do sistema. A gravidade de existência de *jitter* depende

não só da dinâmica da malha de controlo como também da possível perda de amostras. A origem do *jitter* prende-se com diferentes fontes tais como condicionais (dependendo do código) e ciclos variáveis no código, deriva do relógio, algoritmo de escalonamento, etc. Normalmente é modelado como um valor probabilístico uniformemente distribuído (Sanfridson, 2000) mas esta distribuição é muitas vezes inadequada para caracterizar as variações verificadas nos tempos de execução das tarefas.

Conforme referido acima, o escalonamento de tarefas é frequentemente causa de *jitter* devido ao tempo de espera a que as tarefas ficam sujeitas quando concorrem entre si para aceder a recursos partilhados, e.g. o próprio processador. Mesmo reduzido, o jitter pode causar perda de amostras na transmissão entre dois nós, sendo bastante mais crítico em informação binária e controlo lógico que em informação de valores reais (Sanfridson, 2000).

2.1.4. ERROS TRANSITÓRIOS (*TRANSIENT ERRORS*)

Erros transitórios (*Transient Errors*) são definidos como erros introduzidos pontualmente na malha de controlo, causados por defeitos de curta duração (transitórios). Um erro transitório pode ocorrer quando uma amostra é perdida ou corrompida pela rede ou quando interferências eléctricas afectam os sinais da malha de controlo. No caso de perda de amostras o atraso do controlo aumenta e o estado do controlador torna-se inconsistente em relação ao estado do processo. Este tipo de erros pode persistir durante mais do que um período de controlo e, como consequência, se o sistema não recuperar rapidamente poderá apresentar uma falha global. Esta falha global pode ser definida como instabilidade do sistema e deriva da ausência de resultados, ou mesmo de resultados incorrectos, na saída do controlo (Sanfridson, 2000). Pode haver também falhas temporais causadas pelo não cumprimento de *deadlines* críticas. É o caso de quando um controlador não responde suficientemente rápido em relação à dinâmica do processo.

O impacto dos erros transitórios pode ser reduzido seleccionando uma estratégia de escalonamento de tarefas e transmissões apropriada (Sanfridson, 2000). Por exemplo, se se utilizar escalonamento estático, uma vez que se conhece quando cada tarefa ou transmissão deve ocorrer, é relativamente fácil detectar falhas e accionar mecanismos de tolerância a essas falhas. No entanto, devido ao cariz dinâmico dos erros transitórios, os referidos mecanismos de tolerância requerem uma capacidade excedente atribuída estaticamente, logo permanentemente, para que possam funcionar a tempo. Em consequência, quando o sistema opera livre de falhas essa capacidade não é usada, aumentando a ineficiência na utilização dos recursos do

sistema. Por outro lado, o escalonamento dinâmico é, por natureza, flexível e escalável permitindo acomodar alterações *on-line* tais como aquelas necessárias para recuperar de erros transitórios. Essa alterações só serão introduzidas quando necessário, melhorando a eficiência na utilização dos recursos do sistema. Claro está que qualquer erro transitório deverá ser evitado, se possível, no entanto se a divergência do estado do controlador em relação ao processo durante uma falha for reduzida, o respectivo impacto poderá nem sequer ser notado (Sanfridson, 2000). Isto significa que é importante recuperar tão rápido quanto possível dos referidos erros.

2.2. MODELOS DE UM SISTEMA DE CONTROLO DISTRIBUÍDO

Um sistema de controlo distribuído genérico inclui pelos menos três funcionalidades distintas, normalmente designadas por sensor, controlador e actuador. Contudo, existem várias possibilidades de mapear estas funcionalidades sobre os nós físicos do sistema. Trata-se do mapeamento da arquitectura funcional sobre a arquitectura de *hardware*, resultando numa determinada arquitectura operacional. Cada arquitectura operacional tem requisitos de comunicação e computacionais diferentes, resultando em diferentes comportamentos temporais e diferentes desempenhos da malha de controlo.

No entanto, convém referir que por vezes a flexibilidade de configuração da arquitectura operacional é limitada pela natureza do *hardware* e *software* utilizados. De facto, alguns equipamentos são fornecidos de forma fechada para executar apenas uma funcionalidade, não sendo possível adicionar outras. Na última década fizeram-se alguns esforços no sentido de maximizar esta flexibilidade, por exemplo, usando *function blocks* que representam instanciações de funcionalidades genéricas que podem ser adicionadas a equipamento compatível. Nesta secção abordaremos três possíveis mapeamentos.

2.2.1. SENSOR + CONTROLADOR + ACTUADOR

Nesta configuração, o sensor, controlador e actuador encontram-se todos localizados no mesmo nó, figura 2-5. Aqui os problemas com a existência de atrasos induzidos pela rede de comunicação, τ_{sc} e τ_{ca}, referidos na figura 2-4, não se colocam, pois nenhum dos dispositivos se encontra ligado a outro através da mesma. No entanto, colocam-se outros problemas, sendo os de maior relevância, os atrasos associados à execução do algoritmo de controlo, τ_c. Isto é, se no nó várias tarefas (de controlo e outras, e.g., interface de operação, registo de operação, verificação de segurança) competem pelo CPU, como é usual numa aplicação real, poderá surgir um *jitter* no cálculo do sinal de controlo induzido pelo algoritmo de escalonamento das tarefas.

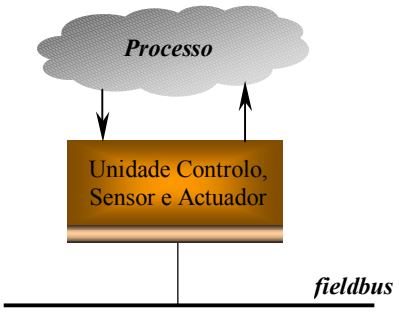

Figura 2-5 – Configuração (Sensor+Actuador+Controlador)

Além do aspecto positivo acima referenciado, a utilização desta configuração torna-se útil em sistemas em que o instante entre amostragem e actuação deverá ser mínimo ou quase nulo.

Outro aspecto a considerar é aquando da existência de malhas de controlo encadeadas onde são usados múltiplos intervalos de amostragem (*multirate sampling*). No exemplo citado acima, controlo da postura e controlo de velocidade de cada uma das rodas de um robô (Magalhães *et al.*, 2002), o uso desta configuração no controlo da velocidade das rodas permite uma elevada redução no tráfego na rede, pois todos os sinais de amostragem e consequente actuação são processados no nó, sendo só enviada uma mensagem para a rede aquando da existência de um *setpoint* por parte do controlador de postura do robô.

É ainda de salientar a flexibilidade que esta configuração apresenta no que respeita à sua instalação junto ao processo.

2.2.2. (Sensor+Controlador)/Actuador ou Sensor/(Controlador+Actuador)

Nestas configurações, a malha de controlo estende-se por dois nós ligados entre si através do barramento de campo. O controlador tanto se pode localizar junto com o sensor como junto com o actuador, num dos nós, enquanto que o actuador ou sensor, respectivamente, se encontra no outro, como está retratado na figura 2-6.

Uma vez que os dois nós se encontram ligados através do barramento de campo, este induz um atraso suplementar devido à comunicação entre eles. No primeiro caso, o atraso induzido pela rede ocorre entre o controlador e o actuador e é referido por τ_{ca}. No segundo caso, o atraso ocorre entre o sensor e o controlador e designa-se por τ_{sc}. Em ambos os casos o atraso depende das características do barramento, da política

de controlo de acesso ao meio (assunto que será abordado mais tarde) e do tráfego existente no barramento. De qualquer modo, existe apenas um fluxo de mensagens envolvido no funcionamento da malha de controlo. O nó que executa apenas uma funcionalidade (sensor ou actuador) pode agora ser muito simples, reduzindo o respectivo custo. Pode-se ainda acrescentar que os nós podem ser colocados em pontos distantes, interligados pelo barramento, de modo a ficarem fisicamente próximos do processo com o qual se relacionam.

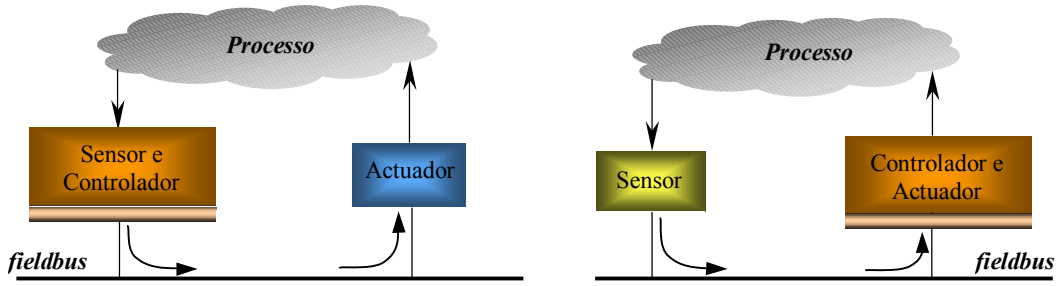

Figura 2-6 – Configurações (Sensor+Controlador) / Actuador e Sensor / (Controlador+Actuador)

No entanto, relativamente à configuração da esquerda, é importante referir o facto de a actuação ocorrer em instantes variáveis, significando que se o processo a controlar depender muito da frescura da amostra adquirida, este atraso poderá comprometer a qualidade do controlo. Este factor, nesta configuração, constitui uma desvantagem bastante relevante fazendo com que não seja adequada no controlo de sistemas deste tipo.

Entretanto na configuração da direita o problema coloca-se no tempo que a amostra demora a chegar ao controlador. Assim poderá ocorrer, na existência de muito tráfego na rede, a perca de amostras (*vacant samples*) entre transmissor e receptor, ou seja, o receptor adianta-se em relação ao transmissor.

2.2.3. Sensor / Controlador / Actuador

O modelo apresentado na figura 2-7 representa outra arquitectura operacional possível, em que a malha de controlo se estende por três nós, cada um dedicado a uma das três funcionalidades referidas: sensor, controlador e actuador. Nesta configuração existem dois fluxos de mensagens que provocam atrasos quer entre sensor e controlador (τ_{sc}) quer entre este e o actuador (τ_{ca}). É, por esta razão, a configuração mais sensível às condições da rede (características, tráfego, etc.) e a que apresenta maior atraso total na malha de controlo, o que torna esta configuração mais adequada para observar o impacto da rede na qualidade do controlo. Por outro

lado, permite isolar as três funcionalidades básicas, facilitando, por exemplo, a respectiva replicação para fins de tolerância a falhas. Apenas o nó controlador poderá necessitar de mais algum poder computacional. O sensor e o actuador poderão ser nós computacionalmente muito simples. Por outro lado, esta configuração também dá uma grande flexibilidade em termos de colocação física dos vários componentes.

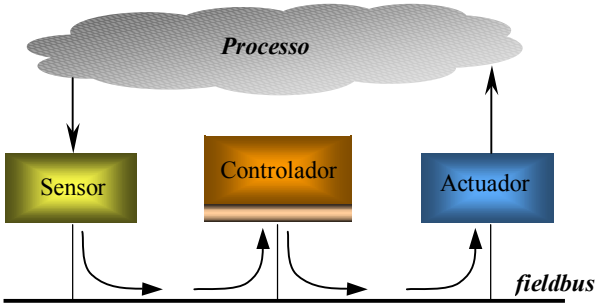

Figura 2-7 – Configuração Sensor/Controlador/Actuador

3. Conclusão

Ao longo deste capítulo foram apresentadas as características gerais de dois tipos de arquitecturas usadas em sistemas de controlo: por um lado a abordagem centralizada e, por outro, os sistemas de controlo distribuídos, caracterizados pela existência de vários nós, interligados por uma rede de comunicação. A comparação e análise destas arquitecturas põem em evidência as vantagens dos sistemas distribuídos em termos de filtragem da informação, modularidade, manutenção da cablagem, tolerância a falhas e escalabilidade.

Foi ainda referido que com o aumento dos níveis de integração electrónica tornou-se possível construir nós com muito poucos componentes. As suas reduzidas dimensões e respectivos custos permitiram instalar arquitecturas distribuídas de controlo mesmo dentro de máquinas ou outros sistemas a controlar, originando os sistemas embutidos distribuídos (*distributed embedded systems*). Assim, um sistema de controlo distribuído embutido contém, tipicamente, uma ou mais malhas de controlo fechadas, constituídas por sensores, controladores, actuadores, etc., que comunicam através de uma rede partilhada (*fieldbus*). A existência de uma rede partilhada possui uma influência determinante no que respeita aos atrasos temporais na malha de controlo. Uma vez que esses problemas temporais condicionam a qualidade do controlo, foi feita uma caracterização dos mesmos num sistema de controlo distribuído de tempo-real.

Por fim foram abordadas as várias possibilidades de mapear as diversas funcionalidades de um sistema de controlo distribuído, designadas por sensor, controlador e actuador, sobre os nós físicos do sistema. Estas possuem requisitos computacionais e de comunicação diferentes, resultando em diferentes comportamentos temporais e diferentes desempenhos da malha de controlo.

Capítulo III

Paradigmas de Controlo Temporal *Time-Triggering* vs *Event-Triggering*

1. Duas Filosofias

Durante muito tempo tem havido discussões entre os defensores de duas filosofias ou paradigmas fundamentais relacionados com o modo como as acções e transacções num sistema são disparadas (Kopetz, 1993). Essas duas filosofias ou paradigmas têm um impacto profundo nas propriedades do sistema global e são designadas por disparo por eventos (*event-triggered*) e disparo por tempo (*time-triggered*) (Rushby, 2001). Estas filosofias estão subjacentes à forma como um sistema é projectado e têm impacto sobre todos os respectivos subsistemas. Assim, como resultado destas filosofias aparecem as arquitecturas *time* e *event-trigger* (Almeida, 2003).

1.1. ARQUITECTURA *TIME-TRIGGERED*

De acordo com o paradigma *time-triggered* toda a comunicação tem lugar em instantes predefinidos ao longo do tempo não tendo em conta variações dos requisitos de comunicação da aplicação em *run-time*. Por esta razão o escalonamento da transmissão tem que ser conhecido de antemão. Isto é facilmente conseguido para a comunicação periódica uma vez que todos os instantes de transmissão futuros podem ser determinados a priori conduzindo a um comportamento temporalmente previsível (*predictability*). Escolhendo um tempo relativo para transmitir (*relative phasing*) entre fluxos (*streams*) de mensagens, a contenção ao nível de acesso ao meio de transmissão pode ser controlada, eliminando ou limitando mútuas interferências induzidas na rede. Esta característica conduz-nos, no que respeita ao comportamento temporal, a uma propriedade importante na gestão de sistemas distribuídos de larga escala e elevada complexidade designada por composabilidade (Kopetz, 1997). Assim, é possível desenvolver subsistemas diferentes e independentes (por exemplo, fabricantes de automóveis e fornecedores), simular com exactidão o comportamento temporal final de cada um e subsequentemente integrá-los no sistema global (Albert, 2004). Outra consequência é um melhoramento do comportamento temporal sob condições de muito tráfego. O controlo desse tempo relativo da transmissão permite um melhor encapsulamento das transmissões do sistema de comunicação conduzindo, nos casos mais críticos, a tempos de resposta mais curtos nos pedidos de comunicação e a uma maior eficiência da utilização da largura de banda (Almeida, 2003). Por último, e não menos importante, o paradigma *time-triggered* potencia uma boa detecção de falhas, uma vez que a falta de qualquer mensagem é imediatamente detectada, permite a eliminação de acessos não autorizados usando *bus guardians*, e ainda facilita a gestão de barramentos redundantes, permitindo detectar facilmente omissões e duplicações (Kopetz, 1997).

Estas propriedades do modelo *time-triggered* são adequadas para aplicações de segurança crítica, como os sistemas *X-by-wire* (*fly and drive by wire*). Por outro lado, a estrutura síncrona das arquitecturas *time-triggered* permite, aos engenheiros, resolver a elevada complexidade dos sistemas de larga escala actuais, tornando-os mais controláveis devido ao maior determinismo já mencionado (Albert, 2004).

Apesar destas propriedades favoráveis, o paradigma *time-triggered* também possui algumas menos favoráveis. Por exemplo, a arquitectura *time-triggered* possui uma escassa flexibilidade. A nível de projecto, é um processo bastante restritivo porque todos os processos e suas especificações temporais devem ser sabidas a priori, caso contrário é impossível uma implementação eficiente. Além disso, a comunicação e o escalonamento das unidades de controlo têm que ser sincronizados durante a operação com o intuito de assegurar rigidamente as especificações do projecto (Albert,

2004). Também no que diz respeito à utilização média da rede, quando parte da informação transportada é gerada de uma forma esporádica e necessita de uma resposta rápida do sistema, por exemplo, alarmes, a arquitectura time-triggered é ineficiente na utilização da largura de banda. Num sistema totalmente *time-triggered*, a transmissão de uma mensagem esporádica necessita da atribuição de uma janela temporal (*time-slot)* periódico que é desperdiçado sempre que não exista mensagem pronta para transmissão (por exemplo, sempre que um alarme não esteja activo) (Almeida, 2003).

1.2. ARQUITECTURA *EVENT-TRIGGERED*

A arquitectura *event-triggered* é baseada na ocorrência de eventos significativos no processo a controlar (Miśkowicz, 2001). Este tipo de arquitectura é mais eficiente que a arquitectura *time-triggered* uma vez que quem envia a mensagem só o faz quando ocorre essa mudança significativa, i.e. um evento. Este tipo de sistemas é mais flexível uma vez que reage prontamente a pedidos de comunicação que podem ser solicitados a qualquer instante de tempo. Por outras palavras, reagem à necessidade de comunicação instantânea (variável). No entanto existem desvantagens na utilização de uma arquitectura *event-triggered*. Esta não suporta composabilidade no que diz respeito ao seu comportamento temporal uma vez que o assincronismo das transmissões dos nós conduz a interferências mútuas sempre que ocorre a integração ou adição de novos nós no sistema. É também necessário ter atenção aos requisitos de comunicação considerando o pior caso (*worst-case*) uma vez que temos que considerar a situação em que todos os nós tentam comunicar simultaneamente. Assim, ou é usada uma maior largura de banda com o propósito de reduzir os tempos de resposta nos pedidos de comunicação ou, então, os tempos de resposta terão de ser maiores. Finalmente uma falha *fail-silent* num nó, por exemplo, causada por um nó se ter desligado, não será imediatamente detectada pelos restantes nós. Para permitir a detecção remota destas falhas, os sistemas *event-triggered* recorrem tipicamente a um mecanismo de *heartbeat* – um mecanismo temporizado usando uma sincronização esparsa (Almeida, 2003).

1.3. *TIME-TRIGGERED* VERSUS *EVENT-TRIGGERED*

É importante referir que a comunicação *time-triggered*, tal como o nome indica, é accionada por tempo. Assim, ao longo de todo o sistema é necessário existir uma noção coerente de tempo. Uma vez que os nós por natureza são assíncronos, devido à dispersão e precisão limitada dos relógios locais, são necessários mecanismos para a sincronização de todo o sistema. Estes, por sua vez, consomem mais largura de

banda embora normalmente pouca. Por outro lado na comunicação *event-triggered*, uma noção global de tempo não é explicitamente necessária, nem mesmo mecanismos para a sincronização de todo o sistema já que cada acção gera os eventos que disparam as acções que compõem cada transacção (Almeida, 2003).

Sob o ponto de vista de eficiência na utilização da largura de banda, o paradigma *time-triggered* é melhor quando se considera uma comunicação periódica, enquanto que para uma comunicação esporádica o paradigma *event-triggered* é mais eficiente. Tipicamente o paradigma *time-triggered* está associado a malhas de controlo enquanto que o *event-triggered* está associado a alarmes e gestão de sistemas. No entanto, vários protocolos suportam uma combinação de ambos os paradigmas (Almeida, 2003; Ferreira *et al.*, 2002), por exemplo, TT-CAN (*Time-Triggered – Controller Area Network*) (ISO/WD11898-4, 2000) e FlexRay (Bogenberger *et al*., 2002). Tal combinação (Ferreira *et al.*, 2002) deverá forçar um isolamento temporal entre ambos os tipos de tráfego (síncrono e assíncrono), caso contrário o assincronismo do tráfego *event-triggered* irá deteriorar o tráfego *time-triggered* devido a mútua interferência. Uma forma típica de alcançar isolamento temporal é construir um mecanismo que não permita a sobreposição das janelas de transmissão de cada um dos tipos de tráfego.

Sob o ponto de vista de segurança, o paradigma *time-triggered* é mais atractivo na construção de sistemas complexos principalmente porque suporta composabilidade no que diz respeito ao comportamento temporal, também devido ao padrão de tráfego predefinido e consequente suporte para uma maior compreensão e verificação do comportamento do sistema e por fim a facilidade na detecção de falhas temporais. Em variadíssimas aplicações de segurança crítica existe uma tendência em direcção aos sistemas *time-triggered*, por exemplo, no campo da aviação com o SAFEbus (Hoyme *et al*., 1992) e na indústria automóvel com o TTP/C (*Time-Triggered Protocol – for class C safety requirements*) (TTTech, 1999), FlexRay e TT-CAN (Almeida, 2003).

À parte as considerações feitas acerca da utilização da rede, é comummente aceite (Thomesse e Chavez, 1999; Peraldi e Dicotignie, 1995; Almeida, *et al*., 2002) que a comunicação usando uma arquitectura *time-triggered* é bem adaptada para aplicações de controlo que tipicamente exigem transmissões regulares de informação com baixo ou mesmo limitado *jitter*. Por outro lado, a comunicação usando uma arquitectura *event-triggered*, é mais adaptada para monitorizar alarmes que são supostos ocorrerem raramente (esporádicos), e tráfego esporádico não tempo-real como por exemplo, a gestão global de um sistema.

2. ARQUITECTURAS *TIME-TRIGGERED* E *EVENT-TRIGGERED* NA PERSPECTIVA DOS SISTEMAS DE CONTROLO DISTRIBUÍDO

2.1. REQUISITOS DE UM SISTEMA DE CONTROLO DISTRIBUÍDO

Nos veículos de gama alta mais recentes, mais de 2500 sinais são trocados por mais de 70 unidades de controlo, fazendo com que os carros hoje em dia sejam considerados um grande sistema de controlo distribuído (Albert, 2004). Com o intuito de reduzir custos de cablagem os nós de um sistema de controlo distribuído são ligados uns aos outros através de uma rede. Os sistemas que usam uma rede de comunicação oferecem a oportunidade de modularidade e escalabilidade, propriedades referidas no capítulo II, as quais oferecem uma maior flexibilidade de implementação de outras variantes. Além disso, sensores e actuadores inteligentes são usados de múltiplas formas para realizar funções e serviços mais complexos.

Infelizmente, as redes de comunicação introduzem atrasos. Além disso, esses atrasos podem variar aleatoriamente perturbando o sistema de controlo.

Se estivermos preocupados em formular os requisitos de um sistema de controlo distribuído duas situações distintas terão que ser colocadas:

1) A operação normal de um sistema de controlo;

2) A ocorrência de uma situação crítica, como por exemplo, o accionamento de um alarme indicando um aumento de pressão numa caldeira.

A segunda situação requer uma reacção, o mais rápida possível, ao evento assíncrono com o objectivo de iniciar mecanismos de emergência.

Os requisitos para a primeira situação surgem a partir da teoria do controlo e podem ser resumidos da forma seguinte:

- O projecto do sistema de controlo, em geral, é suportado na suposição de que os períodos de amostragem são constantes e que o algoritmo de controlo é executado em perfeita periodicidade (Albert, 2004). Se isto não suceder, poderá utilizar-se um algoritmo variante no tempo com o intuito de não colocar em risco o desempenho do controlo ou mesmo a estabilidade do sistema. Tal

algoritmo pressupõe a disponibilidade de alguma informação, como por exemplo, o período de amostragem actual, os instantes a que são realizadas as medidas e o cálculo da variável de controlo ou o tempo exigido pelo algoritmo de controlo.

- Idealmente, os atrasos do sensor e do actuador não existiriam. O primeiro origina um atraso entre a medição da variável do sistema a controlar (*plant*) e a entrada da mesma no controlador. O atraso do actuador deve-se à latência entre o sinal de saída do controlador e a respectiva actuação no sistema a controlar. Uma vez que estes atrasos são inevitáveis, deveriam ser no mínimo constantes com o objectivo de minimizar o *jitter* e assim usar uma lei de controlo invariante no tempo (Franklin *et al*., 1997; Mota *et al.*, 2000). Para os atrasos no sensor, são usados métodos de observação que fornecem o estado das variáveis em instantes bem definidos. Para a compensação dos atrasos no actuador torna-se necessário um projecto de controlador adequado (Albert, 2004; Mota *et al.*, 2000).

- Minimizar os atrasos é sempre vantajoso uma vez que o desempenho global do controlo degradar-se-á com o seu aumento.

2.2. Comparação de Ambos os Paradigmas na Resposta a Eventos Assíncronos

Em sistemas de tempo-real um dos requisitos elementares é a capacidade destes responderem a eventos assíncronos dentro de uma janela de tempo predefinida. Desta forma, pode ser feita uma comparação entre ambos os paradigmas *event-triggered* e *time-triggered*. Tal comparação dá, obviamente, vantagem aos *buses event-triggered*, que transmitem a qualquer momento, relativamente aos *time-triggered* que têm que esperar pela janela própria para transmitir. No entanto a questão que se coloca é qual o preço a pagar para a obtenção de um tempo de resposta adequado em sistemas *time-triggered* para transmitir eventos assíncronos (isto é, qual a largura de banda que irá ser desnecessariamente reservada).

Relativamente a esta questão, (Albert, 2004) refere que uma arquitectura *event-triggered* com pouca sobrecarga do *bus* é mais eficiente uma vez que origina tempos de resposta (latência) e *jitter* pequenos quando reage a tais eventos. No entanto o comportamento degrada-se bastante quando o *bus* apresenta sobrecarga.

Na arquitectura *time-triggered* e para o pior caso, situação mais crítica em que é necessário esperar um ciclo completo, i.e. o evento assíncrono sucede mesmo após o

time slot ter passado (figura 3-1), o comportamento temporal pode ser previamente determinado com e sem sobrecarga do *bus*, ou seja, a arquitectura *time-triggered* é determinística.

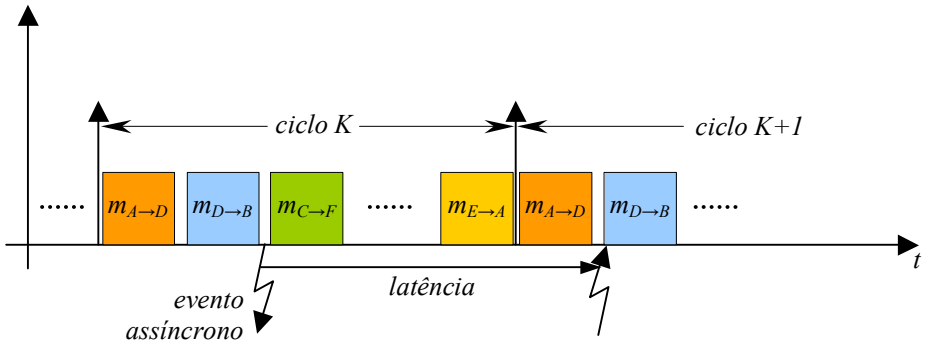

Figura 3-1 – TTbus: ocorrência de um evento assíncrono logo após o *time-slot* ter passado

Como, regra geral, a realidade nem é preta nem branca, tudo dependerá do tipo de aplicação que se pretende implementar, i.e. se se ajusta mais uma arquitectura *time-triggered* ou *event-triggered* ou mista.

2.3. Malha de Controlo Distribuído

Sendo a malha de controlo implementada de uma forma distribuída, com os vários dispositivos espalhados a trocar informação através da rede, figura 2-4 do capítulo II, é conveniente analisar os atrasos entre o amostrador e o controlador e vice-versa, e entre o controlador e o actuador, na perspectiva de ambos os paradigmas, *event-triggered e time-triggered*.

O atraso τ_{sc} define o tempo exigido na comunicação entre o sensor e o controlador, enquanto por τ_{ca} se entende o tempo de atraso entre o controlador e o actuador, como se pode verificar pela figura 2-4 do capítulo II. Ambos os atrasos mencionados dependem essencialmente do tipo de *bus*. Estes atrasos de comunicação são provavelmente a principal razão para a existência de um maior ou menor atraso no controlo e *jitter*, na malha de controlo.

Arquitectura *Event-Triggered*: O atraso no controlo e o *jitter* dependem fortemente da carga do *bus*. Daí, em arquitecturas *event-triggered,* o atraso da comunicação ser variável no tempo e bastante susceptível ao *jitter*.

Um método usado com o objectivo de superar o problema desta variação temporal, proposto por (Luck e Ray, 1990), é o de introduzir *buffers* nos receptores da rede que permitem sustentar as mensagens durante um período fixo igual ao pior caso

de atraso na comunicação. Uma desvantagem deste processo é que o atraso é superior ao mínimo necessário.

Arquitectura *Time-Triggered*: Nas arquitecturas *time-triggered* é essencial sincronizar as acções de todos os nós envolvidos através de um relógio global. Uma vez que o escalonamento define *offline* os *time-slots* para todas as acções, o resultado é um esquema temporal com atrasos potencialmente constantes no controlo (exista ou não carga no *bus*). Se não for implementado um mecanismo de sincronização, o atraso no controlo e o *jitter* poderão ser de magnitude superior à dos sistemas *event-triggered,* pois poderá haver adiantamento do receptor em relação ao transmissor de forma que o primeiro verifica a recepção de uma mensagem mesmo antes da respectiva transmissão tendo de esperar um ciclo até receber esse novo valor.

Outra fonte de atrasos trata-se dos erros transitórios (*transient errors*) (Sanfridson, 2000), referidos no capítulo II. Por exemplo, as mensagens podem-se perder ou corromper. A arquitectura *time-triggered* pode ser mais susceptível aos erros transitórios, uma vez que a retransmissão das mensagens frequentemente não é possível por não ter sido prevista no escalonamento global. Claro está que na arquitectura *time-triggered* a falta de uma mensagem é imediatamente detectada.

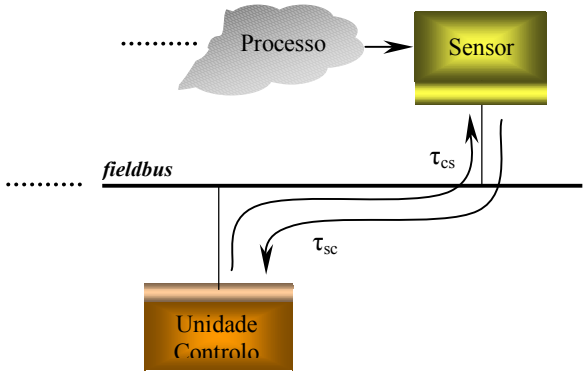

Figura 3-2 – Aquisição de uma amostra pelo controlador

Por outro lado, numa arquitectura *event-triggered* o nó que detecta uma falha de comunicação pode pedir uma retransmissão. Isso provoca mais uma troca de mensagens, por exemplo entre o controlador e o sensor, gerando um atraso suplementar (τ_{cs} e τ_{sc} figura 3-2) mais uma vez dependente da carga da rede.

3. CONCLUSÃO

Existem duas tipologias de projecto de sistemas distribuídos relacionadas com o modo como as acções e transacções são disparadas, nomeadamente *event-triggered* em que os disparos são efectuados após a ocorrência de eventos e *time-triggered* em que os disparos são programados no tempo.

Na generalidade, em sistemas com baixa carga computacional e de comunicações, podemos referir que os sistemas *event-triggered* nos conduzem a atrasos menores. Infelizmente, existem bastantes influências que condicionam os atrasos, fazendo com que estes sejam variáveis no tempo. A sua compensação requer o conhecimento preciso dos instantes reais a que a amostragem tem lugar e a que a respectiva actuação é efectuada, o que nem sempre é possível.

Em contraste, os sistemas *time-triggered* podem conduzir a atrasos maiores mas previsíveis e constantes se as actividades no sistema estiverem todas sincronizadas. Desde que os atrasos surjam do escalonamento, estes são conhecidos a priori e a sua compensação pode ser efectuada através de um algoritmo de controlo constante no tempo.

Capítulo IV

Protocolos de Acesso ao Meio – *MAC Protocols*

1. Introdução

Em qualquer rede baseada em *broadcast* sobre um canal comum acedido por múltiplas unidades ou nós, o problema chave é como determinar quem usa o canal de transmissão sempre que existe disputa por ele, Tanenbaum (2003). Para tornar este ponto um pouco mais claro, consideremos uma teleconferência na qual seis pessoas, em seis telefones diferentes, estão todas ligadas entre si podendo-se ouvir uns aos outros e falar uns com os outros. Imagine-se agora que várias pessoas pretendem falar num determinado instante. Se começarem todas a falar ao mesmo tempo ninguém se entenderá e a comunicação falha. Numa reunião presencial, isso pode ser evitado por meios externos ao canal de comunicação, por exemplo, as pessoas levantam as suas mãos pedindo permissão para falar.

Quando unicamente existe um canal para comunicar, determinar quem transmite a seguir é mais difícil. Este problema é genericamente conhecido como problema do acesso ao meio e são conhecidos muitos protocolos para o resolver, alguns dos quais serão descritos neste capítulo. Na literatura, os canais de *broadcast* são também designados por canais partilhados de múltiplo acesso. Os protocolos usados para determinar quem transmite num canal de múltiplo acesso, pertencem a uma subcamada da camada *Data Link Layer* do modelo *OSI* designada por *MAC* (*Medium Access Control*), retratado na figura 4-1. Os protocolos usados na camada *MAC* podem ser divididos em dois grandes grupos, protocolos de acesso controlado e não controlado (Pleinevaux and Decotignie, 1988; Thomesse, 1998). Dentro do grupo de protocolos de acesso controlado existem ainda duas categorias que são normalmente consideradas: controlo centralizado e distribuído. Inicialmente serão abordados os protocolos de acesso ao meio não controlado e por fim falaremos dos de acesso controlado.

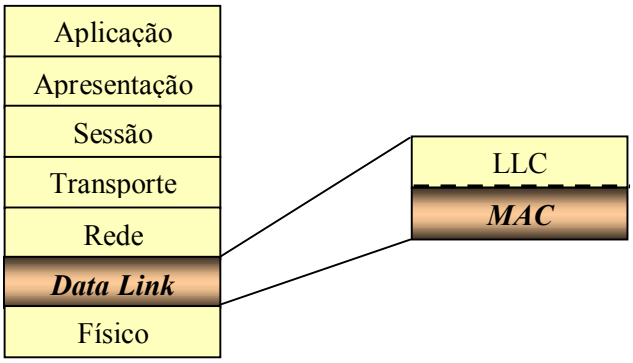

Figura 4-1 – Modelo OSI; subcamadas da camada *Data Link Layer*

2. PROTOCOLOS DE ACESSO NÃO CONTROLADO

2.1. *ALOHA*

Em 1970, Norman Abramson *et al.*, da Universidade do Havaii, projectaram um novo e elegante método de resolver o problema de atribuição do canal de transmissão. A este projecto deram o nome de *ALOHA*. Embora existam já métodos mais evoluídos, e por esse facto este já esteja um ultrapassado, alguns deles resultaram de uma extensão, e melhoramento deste método (Tanenbaum, 2003).

2.1.1. *PURE ALOHA*

A ideia básica do protocolo *ALOHA* consiste em deixar todos os utilizadores (nós) transmitirem sempre que tiverem dados para transmitir, figura 4-2. Deste modo irão ocorrer colisões e as tramas que colidirem serão descartadas. Devido às propriedades de transmissão em *broadcast*, um utilizador que enviou uma trama consegue sempre saber se esta foi ou não descartada escutando o meio. Da mesma forma também os outros intervenientes o conseguem fazer. Se, por aspectos específicos de um dado meio de transmissão, a escuta do meio enquanto se transmite não for possível, então são necessárias confirmações (*acknowledgements – ACK's*) que é o caso das transmissões em *wireless* como em *WiFi*-802.11. Se a trama tiver sido descartada, o nó espera uma quantidade aleatória de tempo e volta a enviar os dados. Este tempo de espera é aleatório para evitar casos em que as tramas colidissem continuamente. Note-se ainda que no *ALOHA* simples um nó não escuta o meio antes de transmitir de modo que não existe forma de saber se outra trama está entretanto a ser transmitida. Este facto leva à possibilidade de ocorrência de colisões mesmo no decurso de transmissões iniciadas com sucesso.

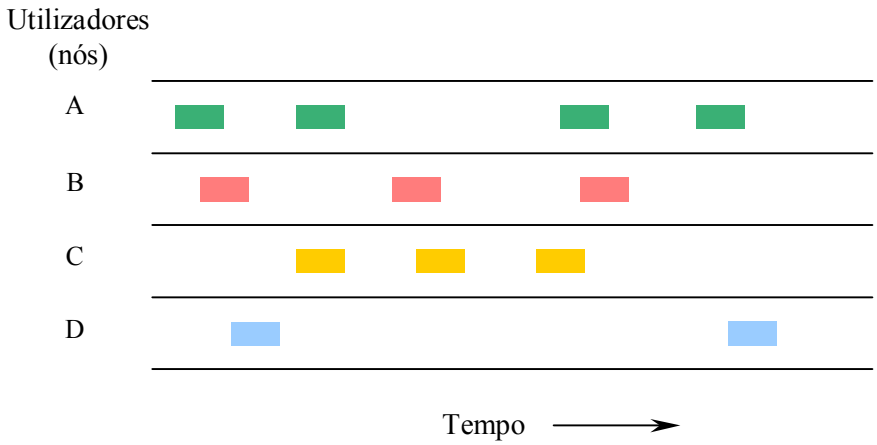

Figura 4-2 – No protocolo *pure ALOHA*, todos os nós transmitem em instantes de tempo completamente arbitrários

2.1.2. *SLOTTED ALOHA*

Em 1972, Roberts publicou um método de duplicação da capacidade do protocolo *ALOHA*. A sua proposta consistia em dividir o tempo em intervalos discretos (*slots*) de dimensões constantes, onde cada um dos intervalos correspondia a uma trama. Este processo requer sincronização e uma forma de a obter seria ter um nó especial que transmitiria um bip no início de cada *slot*, como um relógio.

Neste protocolo as (re)transmissões de tramas iniciam-se sempre no início de cada *slot* e duram exactamente um *slot*. Um nó que queira transmitir uma trama terá de aguardar até ao início do próximo *slot*. Assim, neste método as transmissões ou ocorrem em *slots* disjuntos, logo sem colisões, ou completamente sobrepostas, levando a uma redução da probabilidade de colisões (Tanenbaum, 2003).

2.2. Protocolos de Múltiplo Acesso Através de Escuta do Meio de Transmissão (*Carrier Sense Multiple Access Protocols - CSMA*)

Protocolos em que os nós escutam o meio de transmissão e agem de acordo com o resultado dessa escuta são designados de "**protocolos de escuta do meio de transmissão**" (***Carrier Sense Protocols***). De seguida descrevemos algumas versões deste tipo de protocolos.

2.2.1. *Persistent* e *Nonpersistent CSMA* (*Carrier Sense Multiple Access*)

Quando um nó tem dados para transmitir, primeiro escuta o meio para verificar se existe alguém a transmitir nesse momento. Se o canal de transmissão estiver ocupado, então espera até que esteja livre. Quando um nó detecta o canal livre, começa a sua transmissão enviando a trama ou, se forem várias, a primeira de uma série. Se ocorrer uma colisão, o nó pára de transmitir imediatamente, esperando um tempo aleatório até tentar retransmitir a mesma trama. A este protocolo dá-se a designação de **1-*persistent CSMA protocol***, visto que transmite com probabilidade 1 quando detecta o canal livre (Tanenbaum, 2003).

Quando dois nós pretendem transmitir e existe um terceiro a ocupar o canal de transmissão, estes esperam e, imediatamente após a libertação do canal pelo terceiro nó, começam a sua transmissão originando assim uma colisão. Mesmo assim este protocolo é bastante superior ao *ALOHA* visto que os nós adiam a sua transmissão sempre que o canal está ocupado. Deste modo, as colisões só podem ocorrer no início das transmissões. Assim que a transmissão de uma trama é iniciada com sucesso ela decorrerá livre de colisões até ao fim.

Um protocolo alternativo de escuta do meio de transmissão é designado por ***nonpersistent CSMA protocol***. A diferença fundamental relativamente ao protocolo anterior está relacionada com o comportamento dos nós no caso de pretenderem transmitir e o meio estiver ocupado. Nessa situação os nós não permanecem continuamente a escutar o meio com o objectivo de detectar o fim da transmissão

anterior. Em vez disso, esperam um tempo aleatório antes de voltar a escutar o canal de transmissão. Desta forma este protocolo conduz a uma melhor utilização do canal por menor probabilidade de colisões. No entanto leva a atrasos mais longos que o protocolo 1-*persistent CSMA*.

Outro protocolo deste tipo é o designado por ***p-persistent CSMA*** e é aplicado a canais *slotted*. Quando um nó pretende transmitir escuta o meio. Se estiver desocupado transmite com uma probabilidade p, ou de forma equivalente, a transmissão vai ser retardada para os *slots* seguintes com uma probabilidade q=1-p. Se o canal estiver ocupado, o nó espera que fique livre e ou transmite ou retarda outra vez a transmissão, com as probabilidades p e q, respectivamente. A aplicação das probabilidades p e q faz-se recorrendo a um gerador aleatório de números inteiros num dado intervalo predefinido. Por exemplo, considere-se uma janela de 32 *slots* (0 a 31). A determinação do *slot* efectivo em que uma transmissão vai decorrer pode ser feita gerando um número inteiro aleatório, com distribuição uniforme, entre 0 e 31, inclusive. A transmissão será imediata se o número do *slot* gerado for 0, o que ocorrerá com uma probabilidade p=1/32. Por outro lado, a transmissão será retardada, em relação ao *slot* actual (0), com um probabilidade q=31/32.

Tal como no protocolo anterior, este protocolo também reduz a probabilidade de colisões em relação à versão 1-*persistent* à custa de um aumento do atraso de acesso ao meio. Por esta razão, este tipo de protocolos também aparece referido como um tipo de ***CSMA/CA*** (***CSMA with Collision Avoidance***) embora as colisões não sejam completamente evitadas.

2.2.2. *CSMA* COM DETECÇÃO DE COLISÕES

Os protocolos *CSMA* nas suas versões (1- ou p-) *persistent* e *nonpersistent* representam uma clara melhoria relativamente aos protocolos *ALOHA* porque asseguram que nenhum nó começa a transmitir quando o meio está ocupado. Contudo, é ainda possível realizar outra melhoria, fazendo os nós parar imediatamente a sua transmissão assim que uma colisão seja detectada, poupando tempo e largura de banda. A este tipo de protocolos dá-se a designação de ***CSMA/CD protocols*** (***CSMA with Collision Detection***). Os protocolos *CSMA/CD* são protocolos bem conhecidos no meio das redes de dados locais (*LANs – Local Area Networks*), sendo mais popular o protocolo *Ethernet*.

O funcionamento deste protocolo está esquematizado na figura 4-3. No ponto t_0 um determinado nó acabou de transmitir a sua trama. Outros quaisquer nós que tenham tramas para enviar podem tentar fazê-lo a partir deste momento. Se dois ou mais nós tentarem transmitir simultaneamente irá ocorrer uma colisão.

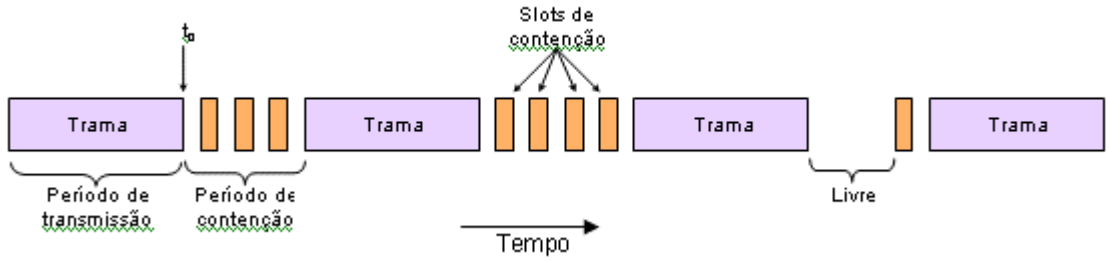

Figura 4-3 – O protocolo *CSMA/CD* pode encontrar-se num de três estados: transmissão, contenção ou livre

Assim que um nó detecta uma colisão, aborta a sua transmissão, espera um período de tempo aleatório (número aleatório de *slots*), e volta a tentar novamente. Dado que poderão existir colisões em cadeia, o protocolo *CSMA/CD* leva a uma alternância entre períodos de contenção e transmissão, com períodos de inactividade (*idle* - livre) que ocorrem quando os nós não têm nada para transmitir.

Relativamente ao período de contenção, é necessário determinar o intervalo de tempo necessário para que um nó tenha a certeza de que pode transmitir, figura 4-4, sabendo que todos os outros não irão interferir. Para determinar esse intervalo vamos assumir o pior cenário. Consideremos τ, o tempo de propagação do sinal eléctrico no canal entre os dois nós mais longínquos. No instante t_0 um nó começa a transmitir. No instante de tempo $t_0+\tau-\varepsilon$, momento imediatamente antes de atingir o último nó, este começa também a transmitir. Este detecta quase instantaneamente a colisão e pára de transmitir, no entanto o outro só irá detectar as perturbações no canal ao fim de um tempo $2\tau-\varepsilon$. Por outras palavras, no pior cenário, um nó só tem a certeza que iniciou a transmissão de uma trama com sucesso se ao fim de 2τ não detectar qualquer colisão. Por esta razão o período de contenção é modelado como um sistema *slotted ALOHA* com a largura do *slot* 2τ.

2.2.3. *The Binary Exponential Backoff Algorithm*

Quando existe a necessidade de retransmissão, devido a colisões, a forma aleatória com que os nós transmitem novamente está condicionada por mecanismos de controlo de retransmissão. Uma possibilidade será, depois da primeira colisão, cada nó esperar 0 ou 1 *slot* de tempo (escolha aleatoria) antes de voltar a tentar transmitir. Se o número que os dois nós escolherem for igual voltarão a colidir (probabilidade de 1/2). Depois da segunda colisão, cada um dos nós escolhe um valor aleatório 0, 1, 2 ou 3, espera esse número de *slots* e volta a tentar. Se uma terceira colisão ocorrer (e a probabilidade de isso acontecer é de 1/4), então o número de *slots* de espera para a

seguinte retransmissão vai ser um valor compreendido entre 0 e $2^3 - 1$, i.e., um valor aleatório entre 0 e 7.

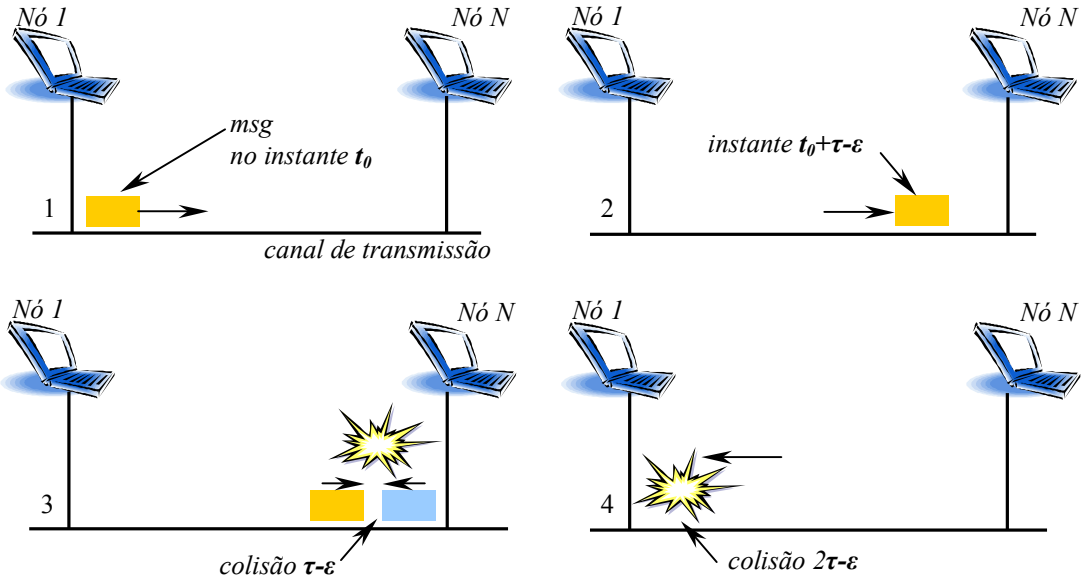

Figura 4-4 – Modelação do período de contenção num sistema *slotted ALOHA*

Em geral, depois de *i* colisões, um número aleatório entre 0 e $2^i - 1$ é escolhido, e esse número de *slots* é o tempo de espera para retransmitir. A este mecanismo dá-se o nome de ***Binary Exponential Backoff* (*BEB*)** Contudo, para evitar atrasos demasiadamente longos, é comum truncar-se o processo a partir de certo número de retransmissões, Por exemplo, Ethernet é um dos protocolos que utiliza o método BEB mas trunca a duplicação do intervalo de valores aleatórios depois de serem alcançadas dez colisões mantendo esse intervalo constante e igual a 1024 *slots*. Também é comum truncar todo o processo de retransmissão ao fim de um certo número de tentativa infrutíferas. No caso de Ethernet esse limite é de 16 tentativas após as quais o nó desiste de tentar transmitir a trama, sendo esta descartada.

2.3. Protocolos Livres de Colisões

Embora no *CSMA/CD* não haja ocorrência de colisões a partir do momento que um nó se apodera do canal de transmissão, estas poderão ocorrer durante o período de contenção. Estas colisões afectam desfavoravelmente o desempenho do sistema especialmente se o cabo é longo (valores grandes de τ) e as tramas são pequenas. Seguidamente iremos ver alguns protocolos que são livres de qualquer colisão mesmo no período de contenção (Tanenbaum, 2003).

Nos protocolos que serão descritos seguidamente, assume-se que existem exactamente N nós, cada um com um endereço desde 0 até N-1, ligados ao canal de comunicação. Note-se que definir a priori o número de nós impõe limitações severas ao nível da escalabilidade de um sistema distribuído no qual se pretende flexibilidade de ligar e desligar nós enquanto os restantes se encontram em funcionamento.

2.3.1. Protocolo Mapeado *Bit*-a-*Bit* (*Bit-Map Protocol*)

Neste protocolo cada período de contenção consiste precisamente em N *slots*, correspondendo cada *slot* a cada nó. Se o nó 0 necessita de transmitir, este coloca um 1 no *slot* 0. Nenhum outro nó tem permissão para transmitir neste *slot*. Da mesma forma o nó 1 obtém autorização para transmitir colocando um 1 no primeiro *slot*. Assim o nó *j* quando pretende transmitir coloca um 1 no *j*-ésimo *slot*. Depois dos N *slots* terem passado, cada um dos nós tem conhecimento de quem pretende transmitir. A partir deste ponto transmitem por ordem numérica. O processo encontra-se descrito na figura 4-5.

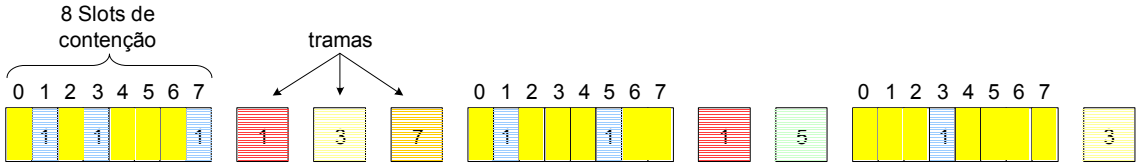

Figura 4-5 – Protocolo *bit-map*

Uma vez que todos sabem quem transmite a seguir, não existirão posteriormente quaisquer colisões. Depois de todos os nós terem transmitido, situação essa que pode ser monitorizada por todos, começa a passar outro *slot* de contenção. Se uma estação estiver pronta a transmitir justamente no momento em que o *slot* tenha acabado de passar, terá de esperar pelo *slot* seguinte (Tanenbaum, 2003). Um princípio semelhante é utilizado no paradigma *Flexible Time-Triggered Communication* (*FTT*) para determinar a transmissão das mensagens síncronas (Almeida *et al*., 2002), mas em que as *slots* de contenção são substituídas por um bit-map enviado por um nó escalonador (*Master*).

2.3.2. CONTAGEM BINÁRIA DECRESCENTE (*BINARY COUNTDOWN*)

O problema do protocolo descrito atrás é que o *overhead* é de um *bit* por nó, o que faz com que não seja muito indicado para redes com milhares de nós. Uma forma de melhorar este modelo, é atribuir a cada nó um endereço binário que terá o mesmo tamanho para todos os nós. Quando um nó pretende usar o canal difunde o seu endereço como uma sequência de *bits*, iniciando essa sequência pelo mais significativo. Todos os *bits* respeitantes a essa posição do endereço, colocados por cada nó, são sujeitos a uma operação de disjunção (*OR*). Este protocolo é por esta razão designado de **Contagem Binária Decrescente** ou ***Binary Countdown***.

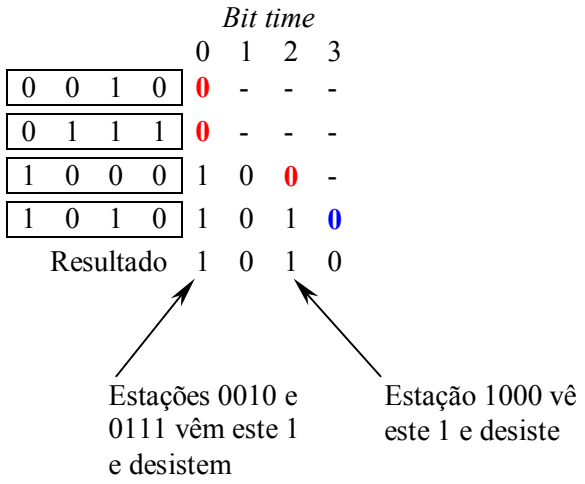

Figura 4-6 – Exemplo de funcionamento do protocolo *binary countdown*

Para evitar conflitos, uma regra de arbitragem é usada: logo que um nó veja que o seu *bit* de maior significância do seu endereço que é 0 foi reescrito com um 1, desiste imediatamente. Para se perceber melhor esta regra, vejamos o seguinte exemplo: se os nós 0010, 0111, 1000 e 1010 pretendem ganhar o canal de comunicação, como se pode verificar pela figura 4-6, transmitem, inicialmente, os valores 0, 0, 1 e 1, respectivamente. Estes *bits* ao serem sujeitos a uma operação de disjunção (*OR*), fornecerão o resultado 1. Os nós 0010 e 0111, ao verem o valor 1, sabem que estão nós de mais elevada prioridade a competir pelo canal de comunicação, pelo que desistem. Agora já só restam os nós 1000 e 1010 e transmitem os valores 0 e 0 (segundo *bit* de mais elevada ordem). Como o resultado da operação é zero, ambas prosseguem. Da mesma forma que se procedeu até aqui, ao transmitirem o terceiro *bit* o resultado é 1. O nó 1000 ao ver esse resultado, desiste. Resta o nó 1010 que ganha o canal pois possui o endereço de ordem mais elevada. A partir deste momento, este nó transmite o que tem a transmitir e, quando finalizar, inicia-se novo processo.

Este protocolo tem a particularidade de que nós com endereços mais elevados têm prioridade relativamente aos nós com endereços mais baixos, o que pode ser bom ou mau, dependendo do contexto onde são implementados. Para aplicações de tempo-real a utilização de prioridades é uma característica útil porque permite determinar os tempos máximos de acesso para cada nó, originando assim um comportamento determinístico (Tanenbaum, 2003).

Este tipo de protocolo também aparece referido como um tipo de ***CSMA/CA*** ou então ***CSMA/DCR*** (***Deterministic Collision Resolution***) ou ainda ***CSMA/BA*** (***Bitwise Arbitration***) e é utilizado, por exemplo, em *Controller Area Network* (*CAN*) mas com os níveis lógicos negados, já que é o nível 0 que ganha a arbitragem (*wired-AND* em vez de *OR*) e é denominado por dominante.

3. Protocolos de Acesso Controlado

3.1. Controlo Centralizado

3.1.1. *Master-Slave*

O controlo centralizado da comunicação tem sido usado desde há bastante tempo recorrendo ao modelo *Master-Slave* (Almeida, 1999). Neste modelo, um único nó, o *Master,* atribui *timeslots* para todos os outros nós da rede, os *Slaves,* poderem transmitir e faz o escalonamento desses *timeslots* de modo a que estes transmitam sem que ocorram colisões, figura 4-7. O *master* endereça um *slave* de cada vez deixando-o transmitir durante uma mensagem. Quando este termina, endereça o seguinte de acordo com o escalonamento e assim sucessivamente.

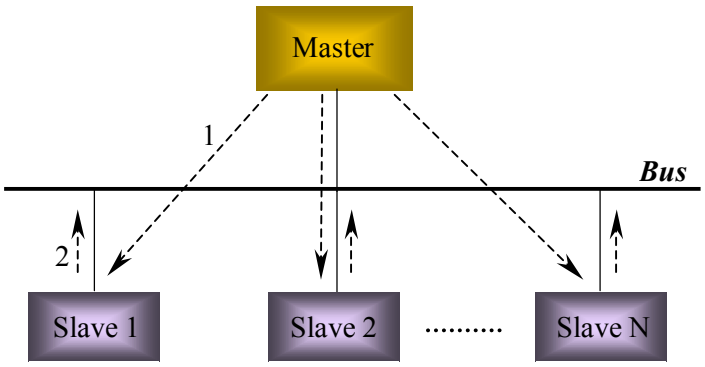

Figura 4-7 – Protocolo de acesso controlado: Centralizado

Trata-se de um modelo bastante fácil de implementar e o comportamento resultante é bastante previsível. Normalmente são sistemas bem ajustados para suportarem comunicações periódicas (*time-triggered*) para as quais é possível determinar antecipadamente todos os instantes de transmissão.

Também se adequam a sistemas com muitos nós muito simples (sensores) que respondem directamente aos comandos do *master*.

3.1.2. PRODUTOR-DISTRIBUIDOR-CONSUMIDOR(ES)

O modelo *Master-Slave* combinado com o modelo Produtor-Consumidor(es) resulta num modelo Produtor-Distribuidor-Consumidor(es) (*PDC*), proposto em (Thomesse, 1993) que é usado no *WorldFIP* (CENELEC, 1996).

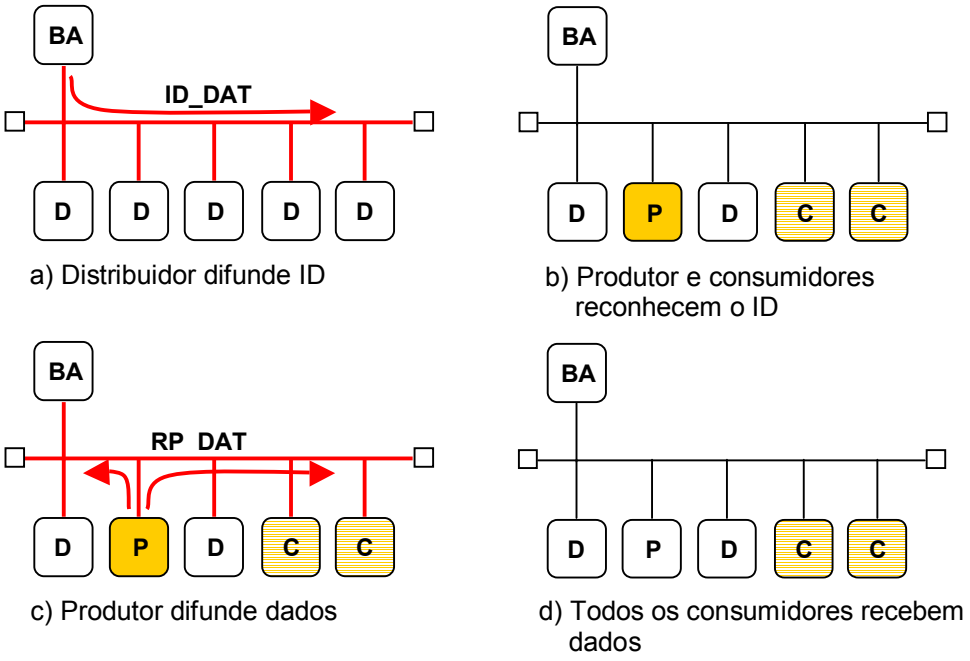

Figura 4-8 – Modelo Produtor-Distribuidor-Consumidor(es)

Neste caso o acesso dos produtores ao *bus* é controlado centralmente por um nó distribuidor (*master*) designado também como árbitro do barramento (*Bus Arbitrator – BA*). O distribuidor (*BA*) endereça uma entidade lógica designada por variável de rede (*network variable*) que tem associados um produtor e mais consumidores, figuras 2-8 a) e b), dá autorização ao produtor para difundir a variável de rede como está retratado na figura 4-8 c) e prepara os consumidores dessa variável para receberem da rede,

figura 4-8 d). Cada par de mensagens de identificação (ID_DAT) e resposta (RP_DAT) é designado por *buffer transfer* e constitui uma transacção periódica. Depois de cada transacção concluída, o *BA* inicia uma nova transacção de acordo com o escalonamento efectuado aquando da configuração do sistema (kim, 2000).

Naturalmente que as entidades terão de possuir identificadores únicos (*ID's*) caso contrário ocorreriam colisões. Note-se que todos os consumidores de uma dada entidade que está a ser produzida recebem a informação do *bus* simultaneamente. Desta forma é suportada consistência espacial da dispersão da informação por todo o sistema. Este modelo é particularmente ajustado para comunicações periódicas (*time-triggered*) ao invés das aperiódicas em que os pedidos de comunicação são gerados pelos nós e é necessário que o *BA* tenha conhecimento deles para os processar. Isto é feito utilizando um bit de sinalização nas mensagens de RP_DAT que indica a presença de um pedido aperiódico.

3.2. CONTROLO DISTRIBUÍDO

3.2.1. *TOKEN-PASSING*

Token-Passing é uma técnica de acesso ao meio através da qual uma mensagem de controlo designada por *token* é passada entre os nós da rede, como se encontra retratado na figura 4-9. Cada nó recebe o *token* e passa-o ao seu vizinho. Enquando detém o *token* o nó está autorizado a enviar dados para a rede

Uma vez que os nós só têm acesso ao *bus* quando possuem o *token*, este não pode ficar em seu poder por um tempo indeterminado. Existe assim a necessidade de que o *token* só permaneça em poder do nó durante um espaço de tempo limitado. Este comportamento temporal é conseguido limitando o tempo que o nó possui o *token*, antes de o enviar para o nó seguinte. Um exemplo bem conhecido é o protocolo *Timed-Token* (Malcolm e Zhao, 1994), o qual foi integrado em alguns tipos de redes incluindo o *token bus* (*IEEE* 802.4) e o *FDDI* (*Fiber Distributed Data Interface*). *PROFIBUS* (DIN, 1990; CENELEC, 1996) usa também uma versão simplificada deste protocolo para controlar o acesso de um conjunto de nós *master* ao *bus*. De salientar que o *PROFIBUS* inclui nós de dois tipos, *master* e *slave*, circulando o *token* unicamente pelos nós *master*. Quando um nó *master* possui o *token*, comunica com os outros nós *master* ou com os nós *slave* que lhe estão associados.

Numa situação em que os nós não respondem devido a falhas, configuração, etc.,este protocolo conduz a uma perca do *token*, causando consideráveis problemas a nível da aplicação. A resolução destes problemas requereria mecanismos de suporte

conduzindo geralmente a paragens na comunicação durante períodos de tempo consideráveis.

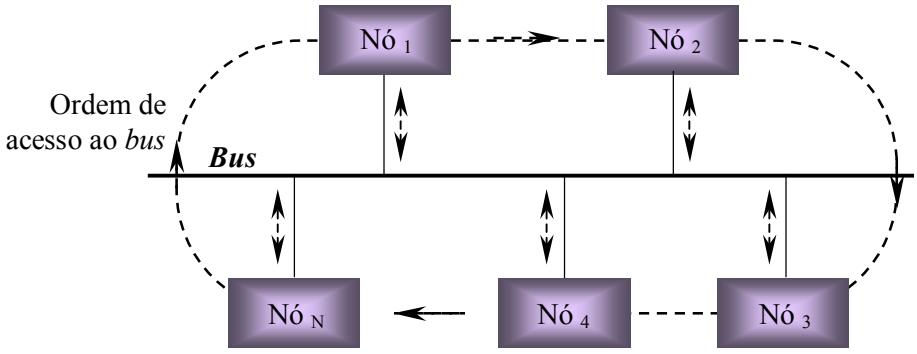

Figura 4-9 – Protocolo de acesso controlado: Distribuído

3.2.2. TOKEN-PASSING VIRTUAL

Outro protocolo de acesso ao meio baseado em *token* é o que é usado no P-Net (DS, 1990; CENELEC, 1996) e no VTPE (Borges, 2003). A ordem pela qual os nós *master* têm acesso ao *bus* é determinada pelo respectivo endereço. Cada nó *master* tem um contador de acessos que é incrementado sempre que ocorre uma transacção bem sucedida ou depois de um *timeout*. Quando o valor do contador de acessos é igual ao endereço de um dado nó *master*, este pode aceder ao *bus*. Quando o valor do contador de acessos ultrapassa o valor do número de nós existentes na rede, valor esse que é especificado na configuração inicial, o sistema recoloca-o a um. Note-se que na realidade não existe propriamente um *token*, mas o protocolo funciona como se existisse, daí a designação de *Token-Passing Virtual*. Este protocolo relativamente ao anteriormente mencionado, é mais robusto pois resolve facilmente a situação de nós que não respondem devido a falhas, configuração, etc. Nesta situação ocorre um *timeout* e todos os nós *master* incrementam os seus contadores de acesso.

3.2.3. *TOKEN-PASSING* CENTRALIZADO

Existe ainda outro protocolo de acesso ao meio baseado em *token* designado por, *Token-Passing* Centralizado. Este é usado no *Foundation Fieldbus-H1* (IEC, 2000; CENELEC, 2000) que combina os acessos ao meio através de um escalonamento centralizado com o tradicional *token-passing*. Cada segmento de rede possui um ou mais *Link Masters*, LMs, onde apenas um realiza o papel de *Link Active Scheduler*, *LAS*. O *LAS* contém uma tabela que indica os instantes (*time-triggered*) de

comunicação de todos os dispositivos que ciclicamente necessitem de transmitir, ou seja, o escalonamento da comunicações periódicas (Almeida, 1999). Por outro lado, o *LAS* concede aos nós a possibilidade de controlar o *bus* usando *token-passing* normal, *pass token* (*PT*) e transmitir mensagens *event-triggered* (não programada) durante janelas temporais precisas que não se sobrepõem às janelas destinadas ao tráfego *time-triggered* (Almeida *et al*., 2002). O *token PT* é circulado por entre os vários nós mas apenas nas janelas respectivas.

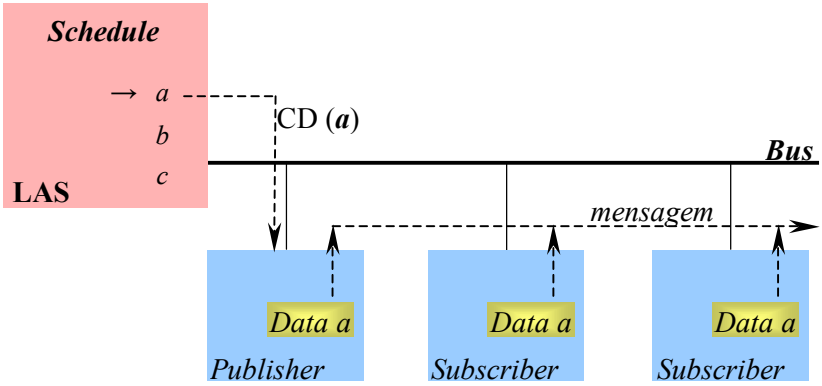

Figura 4-10 – Transferência de dados Programada com *Token Compel-Data* (*CD*)

Existem dois tipos de comunicação programada. A primeira, descrita na figura 4-10, segue o modelo *Publisher-Subscriber* (*PS*) que é semelhante ao modelo *PDC* acima referido, no qual o *LAS* gera uma mensagem *Compel Data* (*CD*) enviando-a para um dado nó produtor (conhecido por *publisher*) de modo a forçá-lo a enviar dados para que os nós consumidores (conhecidos por *subscribers*) a recebam.

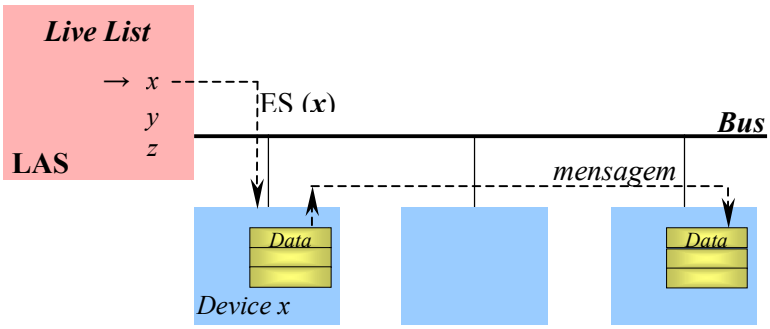

Figura 4-11 – Transferência de dados Programada com Token Execute-Sequence (ES)

Na segunda, descrita na figura 4-11, o *LAS* dá permissão ao *LM* para usar o *bus* durante um período de tempo predefinido, gerando uma mensagem de *Execute Sequence token* (*ES*) (Almeida, 1999). Quando o *LM* recebe o *ES*, tem permissão para

enviar mensagens quer até ter terminado as referências ou terminar o tempo de posse do *token*. As mensagens podem ser enviadas para um único destino (*unicast*) ou vários destinos (*multicast*). No fim, o *LM* deve devolver o *token* para o *LAS*.

3.2.4. TDMA (TIME DIVISION MULTIPLE ACCESS)

Nos protocolos baseados em *TDMA*, cada nó possui um *time-slot* pré-definido onde tem acesso exclusivo ao *bus*. Estes protocolos são particularmente adequados para arquitecturas de segurança crítica, uma vez que facilitam detecção de falhas temporais (Babaglou e Drummond, 1987). Uma característica destes protocolos é que os relógios devem estar sincronizados para garantir que, na difusão de mensagens, não ocorram colisões, caso contrário dois nós poderiam transmitir na mesma janela temporal. Isto requer um mecanismo de sincronização dos relógios, por exemplo baseado na difusão de mensagens (Lönn e Pettersson, 1997; Fonseca e Fonseca, 2000).

Pelo que foi referido anteriormente o controlo de acessos baseado em *TDMA* não necessita de um *token*, uma vez que cada nó sabe precisamente a janela temporal em que vai transmitir (Almeida, 1999). Esta é, por exemplo, a base do protocolo *TTP* (*Time-Triggered Protocol*) (Kopetz and Grünsteidl, 1994; Kopetz, 1997), o qual necessita de um mecanismo de sincronização bastante preciso. A largura das janelas temporais, para acesso ao *bus* por cada nó, pode ser dimensionada a fim de suportar adequadamente tráfego periódico com baixo *jitter* (Almeida, 1999).

Existem duas versões deste protocolo: *TTP*/*C* e *TTP*/*A*. O primeiro usa um algoritmo que realiza a sincronização através de mensagens contendo informação temporal. O último usa uma abordagem de sincronização dos relógios centralizada, i.e., um nó específico inicia cada ciclo *TDMA* através da difusão de uma mensagem de sincronização dos relógios, fazendo com que todos os relógios do sistema fiquem sincronizados pela recepção dessa mensagem. Desta forma todos os nós acedem ao *bus* ordeiramente de acordo com a sua posição no ciclo (Almeida, 1999).

4. CONCLUSÃO

Ao longo deste capítulo foi dada especial ênfase aos Protocolos de Controlo de Acesso ao Meio (*MAC Protocols*) devido ao impacto significativo que geram sobre o comportamento temporal do barramento (*fieldbus*). Vimos ainda que os protocolos usados na camada *MAC* podem ser divididos em dois grandes grupos: protocolos de acesso controlado e não controlado. No primeiro grupo, a forma como os nós acedem

ao barramento é determinada por um controlo externo à aplicação, que pode ser implícito (e.g. tempo ou *token* virtual) ou explícito (e.g. *master poll, token*) e daí facilitarem um comportamento determinístico. No segundo grupo os nós escutam se o meio (barramento) está livre, e só em caso afirmativo é que transmitem. Contudo, ao tentar transmitir podem colidir, sendo sempre necessário utilizar um método de arbitragem, i. é., de resolução de colisões. O método de arbitragem utilizado determina a adaptação do protocolo a sistemas de tempo-real críticos.

Antes de finalizar este capítulo relacionado com os protocolos *MAC*, é importante referir que podem ser feitas outras classificações, além do acesso controlado e não controlado. Em (Malcolm e Zhao, 1995), os protocolos são divididos em duas partes: acesso por controlo da transmissão ou arbitragem. A primeira classificação está estritamente relacionada com os protocolos de acesso controlado onde a linha temporal do *bus* é dividida em janelas temporais (*time-slots*) que são atribuídas aos diferentes nós do sistema. A segunda classificação está relacionada com os sistemas de acesso não controlado onde os atrasos de acesso ao *bus* são regulados por um protocolo de arbitragem distribuída ao nível do *bus* (Almeida, 1999), como por exemplo *CSMA*/*CD*, *CSMA/DCR*, etc.

Capítulo V

Resultados de simulação

1. Introdução

Os sistemas de controlo têm, cada vez mais, vindo a aumentar a sua complexidade quer ao nível das técnicas de controlo quer ao nível da arquitectura computacional usada (Cervin *et al*., 2003).

Actualmente, mesmo os sistemas de controlo embutidos (*embedded*) de reduzida complexidade contêm, frequentemente, um *kernel* multitarefa de tempo-real e suporte de rede. Ao mesmo tempo, o mercado exige também que o custo do sistema seja mantido num valor baixo. Para um melhor aproveitamento dos recursos computacionais, é necessário considerar simultaneamente a implementação do algoritmo de controlo e o software de controlo. Por esta razão, são necessárias novas ferramentas computacionais que facilitem o projecto comum de sistemas de controlo e de tempo-real (Cervin *et al*., 2003).

Muitos sistemas controlados por computador são sistemas distribuídos, constituídos por nós computacionais ligados por uma rede de comunicação pela qual comunicam. É frequente que sensor(es), actuador(es) e controlador(es) residam em diferentes nós, como por exemplo, num robot ou máquina ferramenta. Isto vem dar relevância às malhas de controlo distribuídas, onde dentro de cada nó, e recorrendo a um *kernel* multitarefa de tempo-real, os controladores são implementados como sendo uma ou mais tarefas executando funções específicas.

Na teoria de controlo digital normalmente são assumidos intervalos de amostragem equidistantes e um intervalo entre amostragem e actuação constante ou desprezável. No entanto isto raramente é atingido na prática. Dentro de um nó as tarefas interferem umas com as outras através de preempção ou bloqueio uma vez que os recursos são partilhados. Desta forma os tempos de execução das tarefas em cada nó podem depender das restantes tarefas em execução no sistema, da informação recebida, do ambiente a controlar ou podem variar de acordo com as características do hardware. A nível de um sistema distribuído a comunicação aumenta os atrasos, que poderão ser mais ou menos determinísticos, dependendo do protocolo de comunicação usado (Henriksson *et al*., 2002).

Este determinismo temporal pode ser melhorado através de uma escolha apropriada de técnicas e plataformas de implementação. Por exemplo, a escolha de um método de transmissão com disparo por tempo (*time-triggered*) melhora o determinismo mas tipicamente reduz a flexibilidade operacional, limitando as possibilidades de dinamicamente se realizarem modificações no sistema. Ainda assim, algum nível de indeterminismo temporal é inevitável, associado às variações (*jitter*) dos relógios.

O atraso e respectiva variação (*jitter*) introduzidos pelos sistemas computacionais e pela rede de comunicação podem conduzir a uma significativa degradação do desempenho de um sistema de controlo. Assim, para atingir melhores desempenhos em sistemas com recursos limitados é necessário ter em conta as implementações que se irão adoptar. Para facilitar esta tarefa são necessárias ferramentas de software para simular um sistema de controlo e analisar de que forma os atrasos e *jitter* da informação de controlo afectam o respectivo desempenho.

TrueTime é uma dessas ferramentas que facilita a simulação do comportamento temporal de um *kernel* na execução de tarefas de controlo. Facilita ainda a simulação de modelos simples de redes de comunicação e a sua influência em malhas de controlo distribuído (Cervin *et al*., 2003).

Disponibiliza dois blocos *simulink*: um *kernel* de tempo-real (*Computer Block* ou *TrueTime Kernel*) e um bloco de rede (*Network Block* ou *TrueTime Network*). Os

atrasos na malha de controlo, são capturados por simulação de execução de tarefas no *kernel* e transmissão de mensagens pela rede.

Numa perspectiva de tempo-real, é necessário primeiro identificar os constrangimentos temporais (*deadlines*, períodos, *offsets*,...) das diferentes tarefas do sistema, bem como das respectivas comunicações, e depois é necessário proceder ao escalonamento dos CPU's e da rede de forma que todos os constrangimentos sejam cumpridos durante a execução.

2. AMBIENTE DE SIMULAÇÃO USANDO O TRUETIME

Como já foi referido anteriormente, o *TrueTime* possui dois blocos *simulink* base, um *kernel* e um bloco de rede, de acordo com a figura 5-1. O *kernel* é accionado por eventos (*event-driven*) e executa tarefas definidas pelo utilizador e interrupções representando, por exemplo, tarefas de I/O, algoritmos de controlo e interfaces de rede. A política de escalonamento de cada *kernel* é arbitrária e decidida pelo utilizador. Da mesma forma, no bloco de rede, as mensagens são enviadas e recebidas mediante o modelo de rede escolhido que poderá ser por exemplo, *CSMA*/*CD*, *CAN*, *Token Bus*, *TDMA*, etc., (Cervin *et al*., 2003).

Ambos os blocos são accionados por eventos (*event-driven*) internos e externos. Exemplos de eventos internos são, um *timer* que expirou, a realização completa de uma tarefa ou a transmissão integral de uma mensagem. Eventos externos correspondem a interrupções externas como, a chegada de uma mensagem da rede. O bloco *kernel* executa tarefas e rotinas de serviço às interrupções (*ISR's –Interrupt Service Routines*) definidas pelo utilizador. Entretanto, o bloco de rede, envia e recebe mensagens de acordo com o modelo de rede definido pelo utilizador.

O simulador permite modelar o tempo de execução das tarefas e a transmissão de mensagens como constante, aleatório ou dependente da informação (*data dependent*). Esta parametrização pode ser usada com diferentes objectivos (Cervin *et al*., 2003):

- Para investigar os efeitos do indeterminismo temporal causado por preempção ou bloqueio nas tarefas ou nas transmissões;
- Para desenvolver esquemas de compensação que ajustem dinamicamente o controlador com base em variações temporais actuais;
- Para testar abordagens novas e mais flexíveis de escalonamento dinâmico;

- Para simulação de sistemas de controlo *event*-driven.

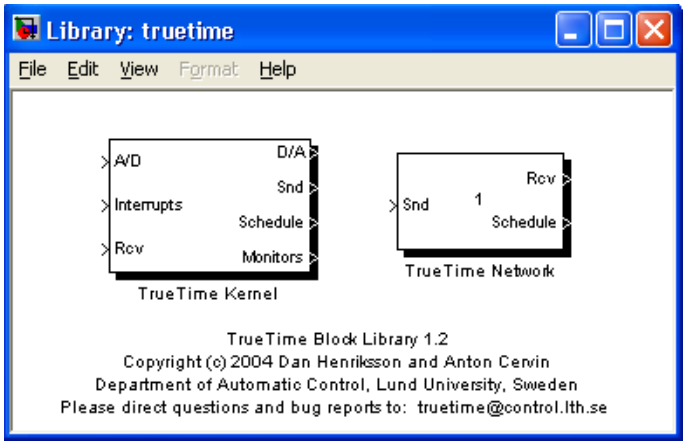

Figura 5-1 – Blocos *Kernel* e *Network* do *TrueTime*

2.1. *COMPUTER BLOCK (KERNEL)*

O bloco computacional simula um computador com um *kernel* multitarefa simples e flexível, conversores A/D e D/A, interface de rede e canais de interrupção externa, figura 5-1.

Internamente o *kernel* mantém várias estruturas de dados típicas em *kernels* de tempo-real: uma fila de tarefas prontas a executar, uma fila de tarefas a aguardar activação, descritores de tarefas, rotinas de atendimento a interrupções, monitores e temporizadores, as quais foram criadas para suportar a simulação.

A execução das tarefas e interrupções é definida através de código fornecido pelo próprio utilizador. Estas funções podem ser escritas em C++, para maior rapidez, ou para maior facilidade de utilização, em código MATLAB (*m-files*).

2.1.1. TAREFAS

A tarefa é o principal elemento construtivo no ambiente de simulação do *TrueTime*. Permite simular actividades periódicas, tais como tarefas de I/O e controlo, e aperiódicas, como tarefas de comunicação e controladores accionados por eventos (*event-driven*).

Um número arbitrário de tarefas pode ser criado para ser executado pelo *kernel*. Cada tarefa é definida por um nome (nome da função) e um conjunto de atributos. Os atributos incluem o nome, tempo de execução efectivo, tempo de execução no pior

caso (*wcet*), tempo de execução acumulado (*budget*), *deadlines* relativas e absolutas, prioridade (se é usado escalonamento com prioridade fixa) e período (se a tarefa é periódica). Alguns dos atributos, tais como o tempo de execução e *deadline* absoluta, são constantemente actualizados pelo *kernel* durante a simulação. Outros, como o período e a prioridade, são normalmente mantidos constantes podendo ser alterados através de invocações às primitivas do *kernel* enquanto a tarefa é executada.

2.1.2. SUPORTES DE INTERRUPÇÃO

As interrupções podem ser geradas de duas formas: internamente ou externamente. Uma interrupção externa está associada a um canal de interrupção externa do *kernel*. A interrupção é accionada quando o sinal na entrada do canal de interrupção muda o valor. Este tipo de interrupções pode ser usada para simular a chegada de uma medida pela rede de comunicação em sistemas de controlo distribuído.

As interrupções internas estão associadas a temporizadores (*timers*). Podem ser criados temporizadores periódicos, onde cada vez que o *timer* expira a *ISR* associada é activada e escalonada para ser executada ou temporizadores por disparo único (*one-shot timers*), onde quando o *timer* associado expira é executada, uma única vez, a ISR associada. Quando uma interrupção interna ou externa ocorre, uma *ISR* é escalonada para servir a interrupção. Uma *ISR* funciona da mesma forma que uma tarefa, mas é escalonada com um nível de prioridade mais elevado.

Uma interrupção é definida por um nome, prioridade e nome da função que lhe está associada. As interrupções externas também têm o atributo latência, que significa que durante esse intervalo de tempo a interrupção é insensível a novas invocações.

2.1.3. CÓDIGO

O código associado às tarefas e interrupções é escalonado e executado pelo *kernel* à medida que a simulação avança. O código é normalmente dividido em vários segmentos, como descrito nas figuras 5-2 e 5-3. Este pode interagir com outras tarefas e com o ambiente no início de cada segmento. Este modo de execução torna possível a modelação de atrasos de I/O, bloquear quando se acede a recursos partilhados, etc. O tempo de simulação de cada segmento de código é devolvido pela função e pode ser modelado como constante, aleatório ou *data dependent*. Durante a simulação, o *kernel*, executa as funções associadas a cada segmento em sequência respeitando os intervalos de tempo especificados. Para além disso, o *Kernel* guarda a informação de cada segmento em cada instante, permitindo que cada tarefa seja interrompida para execução de tarefas de maior prioridade (preempção) e retomada mais tarde. Assim,

no caso de preempção, é possível que o tempo actual entre segmentos seja superior ao especificado.

```
function [exectime, data] = sens_code(segment, data)

switch segment,
  case 1,
     data.y = ttAnalogIn(1);     % Lê valor analógico do processo
     exectime = 0.0004;
  case 2,
     ttSendMsg(4, data.y, 10);  % Envia mensagem para nó 4 (controller)
     exectime = 0.0007;
  case 3,
     exectime = -1;              % Terminou
end
```

Figura 5-2 – Exemplo de uma função de código

A figura 5-2 mostra um exemplo de código de uma tarefa, composta por 3 segmentos, que implementa um sensor remoto. O primeiro segmento amostra o processo, demorando 0,4ms. Assim que a amostragem termina, o segundo segmento é invocado para enviar uma mensagem contendo o sinal obtido pelo sensor, demorando 0,7ms. O terceiro segmento indica a finalização da tarefa, sinalizada através de um tempo de execução negativo. A linha temporal correspondente à execução desta tarefa é ilustrada na figura 5-3.

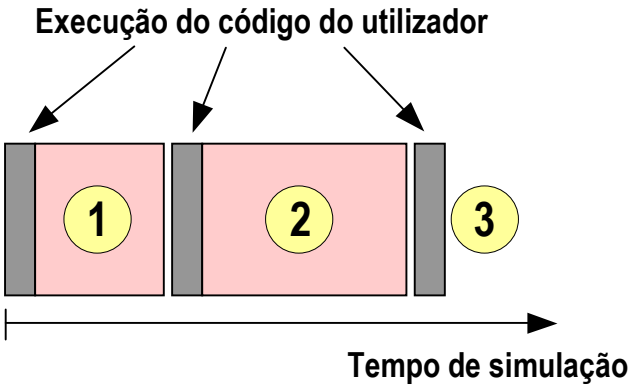

Figura 5-3 – A execução do código associado às tarefas e rotinas de interrupção é modelado através de segmentos com diferentes tempos de execução (1, 2 e 3)

A estrutura `data`, representa a memória local da tarefa e é usada para guardar o sinal amostrado. A conversão A/D e envio de mensagens é realizada usando as primitivas do *kernel*, `ttAnalogIn` e `ttSendMsg`.

Além da conversão A/D e envio de mensagens pela rede, existem outras primitivas que podem ser invocadas a partir de funções. Estas incluem funções de conversão

D/A, receber mensagens, criar ou remover temporizadores, criar operações de monitorização, e alterar atributos das tarefas. Algumas das primitivas do *kernel* estão listadas na tabela 5-1.

`ttAnalogIn(ch)`	Obtém o valor de um canal de entrada
`ttAnalogOut(ch, value)`	Aplica o valor ao canal de saída
`ttSendMsg(rcv, data, len)`	Envia mensagem para a rede
`ttGetMsg()`	Recebe mensagem da rede
`ttSleepUntil(time)`	Espera até passar um tempo específico
`ttCurrentTime()`	Tempo actual de simulação
`ttCreateTimer(name, time, ih)`	Dispara uma interrupção num instante específico
`ttEnterMonitor(mon)`	Activa um monitor
`ttWait(event)`	Espera por um evento
`ttNotifyAll(event)`	Activa todas as tarefas que estejam à espera de um evento
`ttSetPriority(value)`	Altera a prioridade de uma tarefa
`ttSetPeriod(value)`	Altera o período de uma tarefa

Tabela 5-1 – Exemplos de primitivas do *kernel* que podem ser invocadas a partir de funções associadas a tarefas ou interrupções

2.1.4. GERAÇÃO DE GRÁFICOS DE SAÍDA

O *kernel* gera dois gráficos provenientes das saídas monitor e *schedule*. O primeiro mostra que tarefa(s) estão inactivas e em espera nos diferentes monitores durante a simulação. A geração destes gráficos é opcional e pode ser especificada individualmente para cada tarefa, suporte de interrupção e monitor. O último mostra graficamente a execução das tarefas e das *ISR's* ao longo da simulação, usando um gráfico de Gantt. Para cada uma são utilizados três níveis. O nível alto significa que essa tarefa ou *ISR* está a ser executada. Um nível médio indica que a tarefa ou *ISR* está em espera (*preempted* ou *blocked*), enquanto que um nível baixo mostra que a tarefa ou *ISR* não está activa (*idle*). De salientar ainda que as *ISRs* possuem prioridade em relação às tarefas.

2.2. *NETWORK BLOCK* (BLOCO DE REDE)

O *Network Block* ou Bloco de Rede é, também, *event-driven* e é accionado quando ocorre passagem de mensagens. Uma mensagem contém informação acerca do nó que a envia e do que a recebe, dados arbitrários do utilizador (tipicamente medidas ou

sinais de controlo), o comprimento da mensagem e atributos opcionais de tempo-real, como prioridade ou *deadline*.

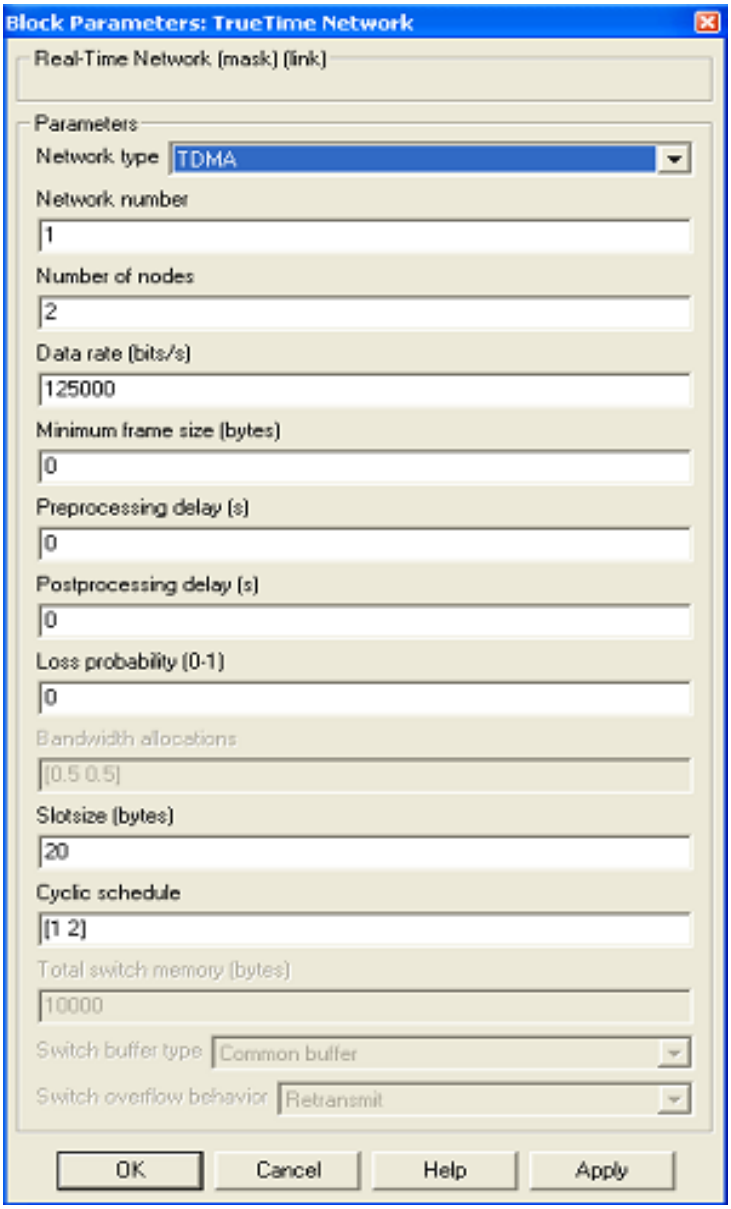

Figura 5-4 – Janela de parâmetros do bloco de rede

No bloco de rede, é possível especificar a taxa de transmissão e o protocolo de acesso ao meio (*CSMA/CD*, *CSMA/AMP*, *FDMA*, *TDMA*, *Round Robin*), entre outros parâmetros, como se pode verificar pela figura 5-4. Uma mensagem pode ser fraccionada em tramas mais pequenas que são transmitidas em sequência, cada uma com um cabeçalho adicional específico ao protocolo utilizado. Quando termina a

simulação da transmissão de uma mensagem, esta é colocada num *buffer* do nó de destino, o qual é avisado por uma interrupção de *hardware*.

Da mesma forma que o bloco de *Kernel*, o bloco de rede também gera graficamente através da saída *schedule* o gráfico de Gantt das transmissões de mensagens através da rede, usando uma codificação semelhante dos estados de transmissão, espera e inactivação.

3. SIMULAÇÕES EFECTUADAS

Nas secções seguintes serão realizadas experiências que visam compreender melhor o impacto das redes de comunicação em sistemas de controlo distribuído. Para tal serão criados cenários de teste que incidirão essencialmente sobre os modelos referidos na secção 2 do capítulo II, tendo como base de trabalho a ferramenta de simulação *TrueTime*.

O primeiro cenário tem como objecto de estudo o comportamento da malha de controlo quer sem o uso de rede de comunicação (modelo sensor, controlador e actuador num único nó), quer com rede a interligar os vários nós. A primeira situação serve como referência de melhor desempenho do controlo. A segunda e terceira, usando dois nós ou três nós respectivamente, irão servir para analisar o impacto dos atrasos induzidos pela rede de comunicação na qualidade de controlo, através da medição do erro médio quadrático. Em ambas as simulações será usado o protocolo de acesso ao meio (*MAC*) *CSMA*/*AMP* (utilizando em *CAN*), já descrito no capítulo III, com uma taxa de transmissão de 125 kbps e tamanho das mensagens de 10 bytes (tamanho total da trama incluíndo campos de dados e de controlo).

O segundo cenário de simulação consistirá no estudo do modelo em que os dispositivos, sensor, actuador e controlador, se encontram dispersos pela rede (modelo totalmente distribuído). Este cenário conterá, numa primeira experiência, um nó de interferência com prioridade mais elevada relativamente aos nós da malha de controlo, que enviará um fluxo de mensagens através da rede, com características estocásticas mas taxa de utilização média da rede relativamente baixa. Numa segunda experiência será aumentado o nível de interferência na rede por esse nó, aumentado a respectiva taxa de utilização média da rede. Uma outra experiência irá consistir em adicionar mais um nó à rede, mantendo os restantes parâmetros inalterados, mas com baixa prioridade. Uma última experiência consistirá em colocar ambos os nós de interferência com prioridades mais elevadas que os nós da malha de controlo. Todas

as simulações realizadas usarão o protocolo acima referido, à mesma taxa de transmissão e com o mesmo tamanho da mensagem.

No terceiro cenário serão efectuadas as mesmas experiências que foram realizadas para os cenários anteriores mas usando diferentes *MAC's*, nomeadamente o protocolo *CSMA*/*CD, Round Robin* e *TDMA*. No final do capítulo é feita uma análise dos resultados obtidos por simulação para os diferentes cenários.

3.1. Cenário 1 – Distribuição (*CSMA/AMP*)

Este primeiro cenário de simulação consiste em testar o desempenho do controlo sobre um processo tendo em consideração os diferentes modelos apresentados no capítulo II.

3.1.1. Modelo Sensor + Controlador + Actuador

Nesta configuração, o sensor, o controlador e o actuador encontram-se todos localizados no mesmo nó como mostra a figura 2-5 (capítulo II). Uma vez que não existe rede a interligar ambos os dispositivos, os atrasos inerentes à mesma não irão ocorrer.

Em todas as simuações é usado um controlador PD (proporcional diferencial) simples para controlar um processo constituído por um servo *(DC-servo)*. O processo é controlado através de uma tarefa implementada num bloco *kernel* do *TrueTime*.

O controlador PD foi implementado de acordo com as seguintes equações:

$$P(k) = K \cdot (r(k) - y(k))$$
$$D(k) = a_d D(k-1) + b_d (y(k-1) - y(k))$$
$$u(k) = P(k) + D(k)$$

onde $a_d = \frac{T_d}{N \cdot h + T_d}$ e $b_d = \frac{N \cdot K \cdot T_d}{N \cdot h + T_d}$ (Åström e Hägglund, 1995).

Desta forma o modelo da figura 5-5, apresentado em ambiente do *MATLB*/*Simulink* usando as *toolboxes* do *TrueTime*, será o que irá apresentar um melhor desempenho de controlo.

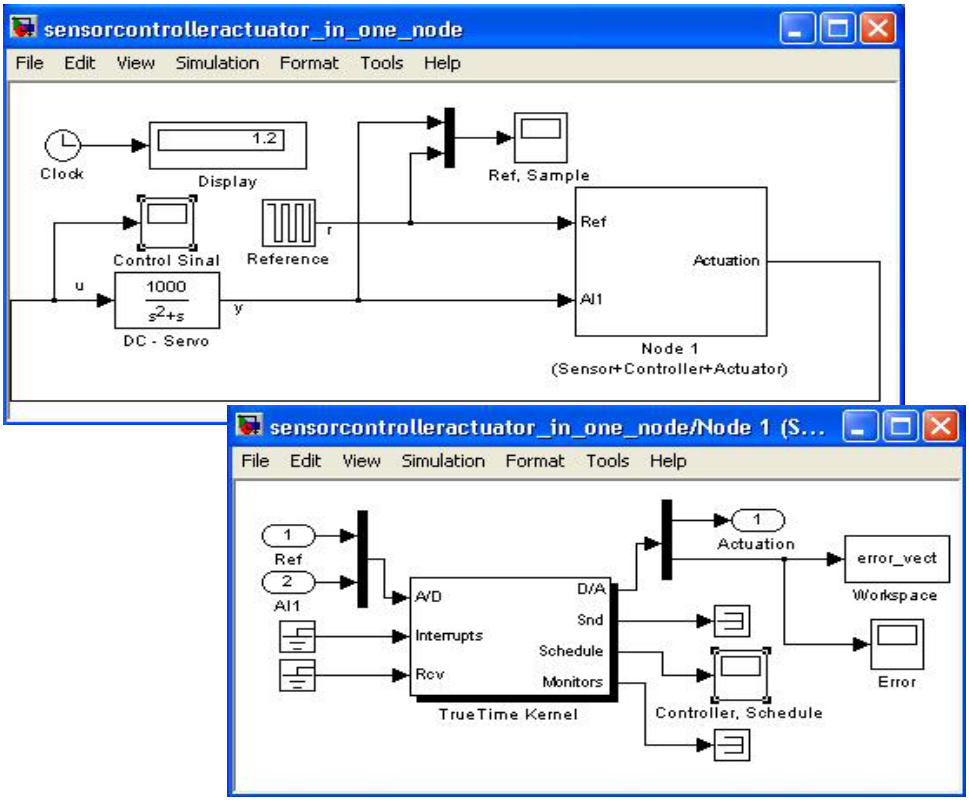

Figura 5-5 – Implementação do modelo Sensor+Controlador+Actuador num só nó usando o simulador *Truetime*

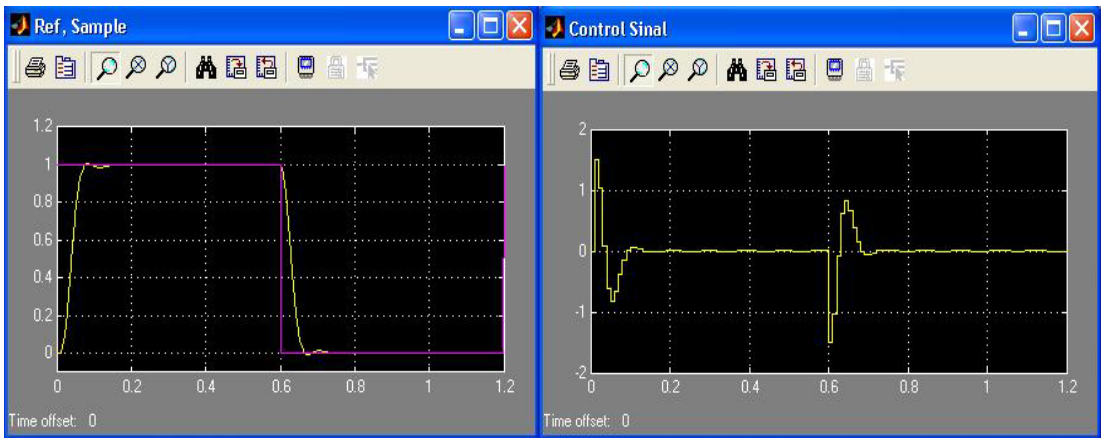

Figura 5-6 – Sinais de amostragem e de referência (esquerda) e de controlo (direita)

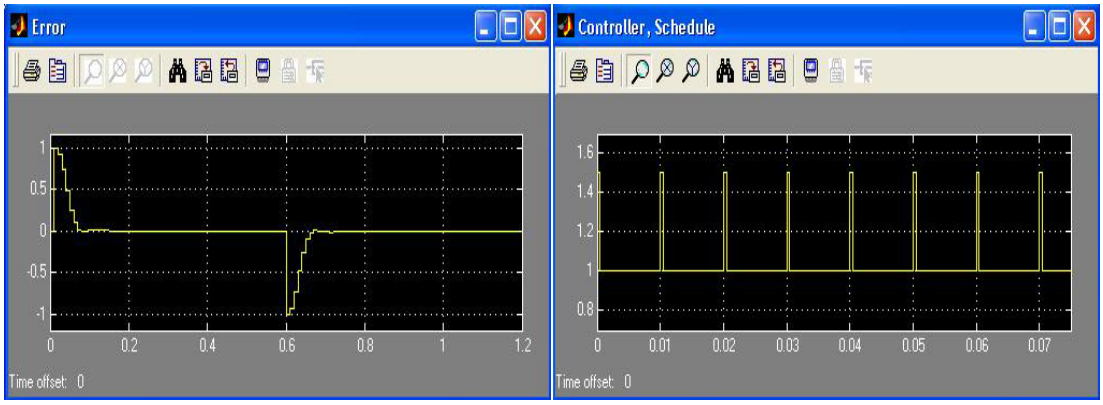

Figura 5-7 – Erro médio quadrático (esquerda) e escalonamento do sinal de controlo (direita)

Na figura 5-6 são apresentados os sinais de amostragem e de referência (esquerda), bem como o sinal de controlo (direita). A figura 5-7 mostra, graficamente, a evolução do erro (esquerda) e o escalonamento dos instantes em que é feito o controlo (direita). Para uma melhor compreensão, refira-se que o controlo é efectuado com período de 10*ms* através da execução de uma tarefa que lê o valor do processo, determina o sinal de controlo e actua sobre o processo. A figura 5-8 mostra o código em *Matlab* usado para criar a tarefa de controlo e estabelecer os parâmetros da malha.

```
function sens_act_ctrl_init

% Inicializaçao do Kernel do True Time
ttInitKernel(2, 2, 'prioFP');

% Parametros do controlador
h = 0.010;          % período de amostragem
N = 100000;         % número de amostras
Td = 0.035;
K = 1.5;            % ganho do controlador

% Criar tarefa de dados (memória local) para um controlador PD
data.u = 0.0;
data.K = K;
data.ad = Td/(N*h+Td);
data.bd = N*K*Td/(N*h+Td);
data.Dold = 0.0;    % D(k-1)
data.yold = 0.0;    % y(k-1)

% Criar tarefa do controlador
deadline = h;
prio = 2;
offset= 0;
ttCreatePeriodicTask('ctrl_task', offset, deadline, prio,
                     'sens_act_ctrl_code', data);
```

Figura 5-8 – A execução do código associado com a malha de controlo no modelo Sensor+Controlador+Actuador

3.1.2. MODELO (SENSOR+CONTROLADOR)/ACTUADOR

Nesta configuração, a malha de controlo estende-se por dois nós ligados entre si através do barramento de campo (rede). O controlador tanto se pode localizar junto do sensor como junto do actuador. No nosso caso, o sensor encontra-se junto do controlador, encontrando-se o actuador num nó à parte.

Com esta simulação pretende-se analisar a degradação do desempenho do controlo do mesmo processo e do mesmo controlador quando se utiliza uma mensagem na malha de realimentação, sem mais nenhuma perturbação. Os diagramas Simulink associados a este sistema estão ilustrados na figura 5-9.

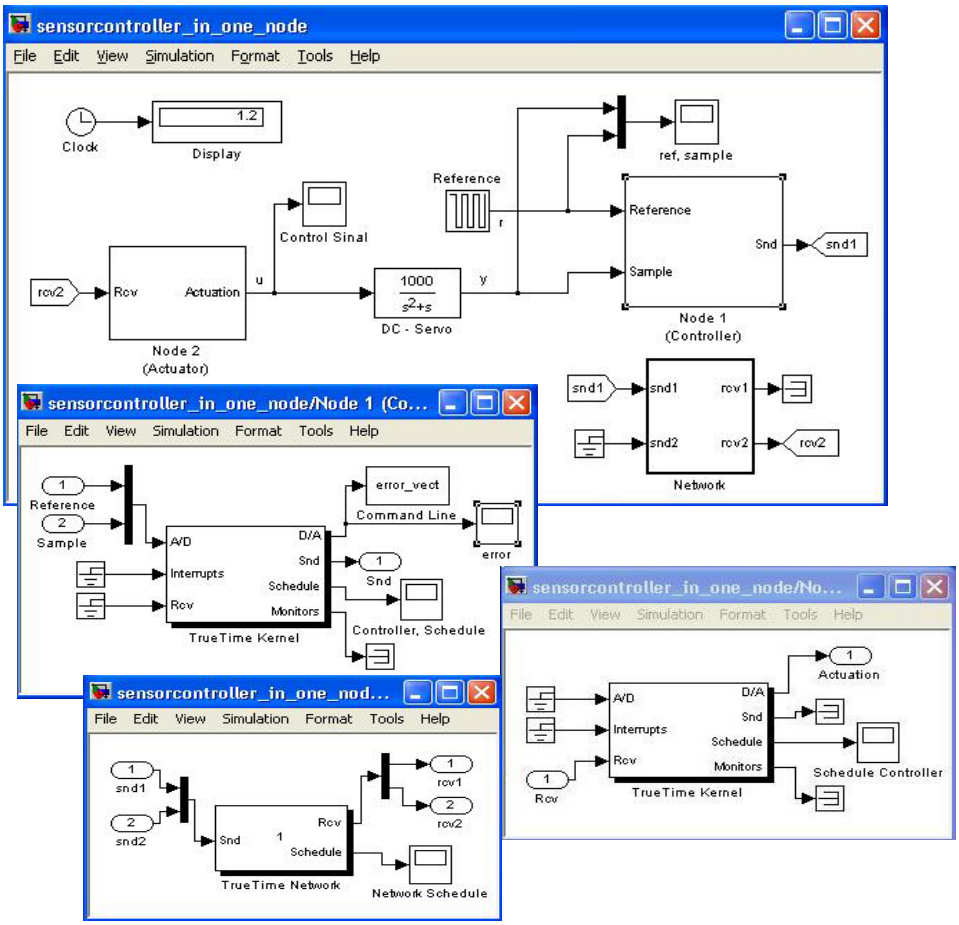

Figura 5-9 – Implementação do modelo (Sensor+Controlador) / Actuador usando o simulador *Truetime*

Na figura 5-10 estão representados os sinais de amostragem e de referência (cima), que indicam um ligeiro aumento do tempo de estabelecimento (*settling time*), e o escalonamento das mensagens (baixo) enviadas através da rede pelo nó 1 (sensor e controlador). Uma vez que o nó 2 (actuador) não transmite mensagens visualiza-se uma linha contínua.

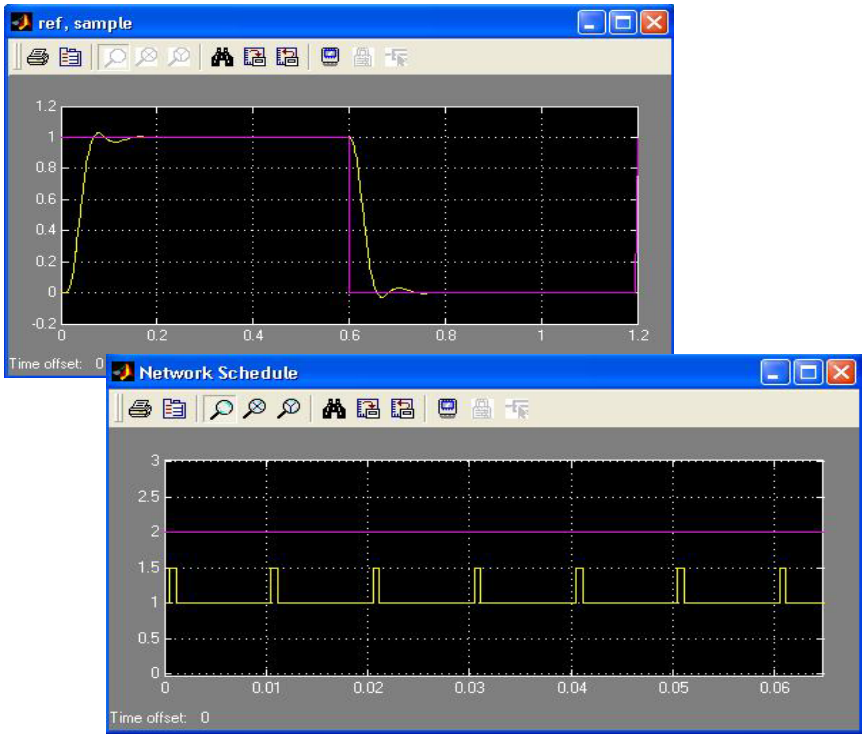

Figura 5-10 – Sinal de amostragem e de referência (em cima) e escalonamento de mensagens da rede (em baixo) para o modelo (Sensor+Controlador)/Actuador usando o protocolo de comunicação *CSMA/AMP*

3.1.3. Modelo Sensor/Controlador/Actuador

Nesta configuração usam-se dois fluxos de mensagens que aumentam os atrasos quer entre sensor e controlador quer entre este e o actuador. É, por esta razão, a configuração que apresenta maior atraso total na malha de controlo, o que a torna mais adequada para observar o impacto da rede de comunicação na qualidade do controlo em sistemas distribuídos sob diferentes condições. O respectivo modelo *Simulink* está retratado na fiura 5-11.

Na figura 5-12 estão representados graficamente os valores do sinal de referência e amostragem do sistema (cima). É visível um aumento da oscilação (*ringing*) em torno dos valores de equilíbrio, com uma pequena sobrereacção (*overshoot*) e aumento do tempo de estabelecimento (*settling time*), efeitos estes que resultam dos atrasos inerentes à rede. A figura 5-12 mostra também o escalonamento das mensagens na rede (baixo). Podemos identificar a existência de 3 nós, onde apenas o sensor e o controlador enviam mensagens, com período de 10*ms*, igual ao intervalo de amostragem, *h*.

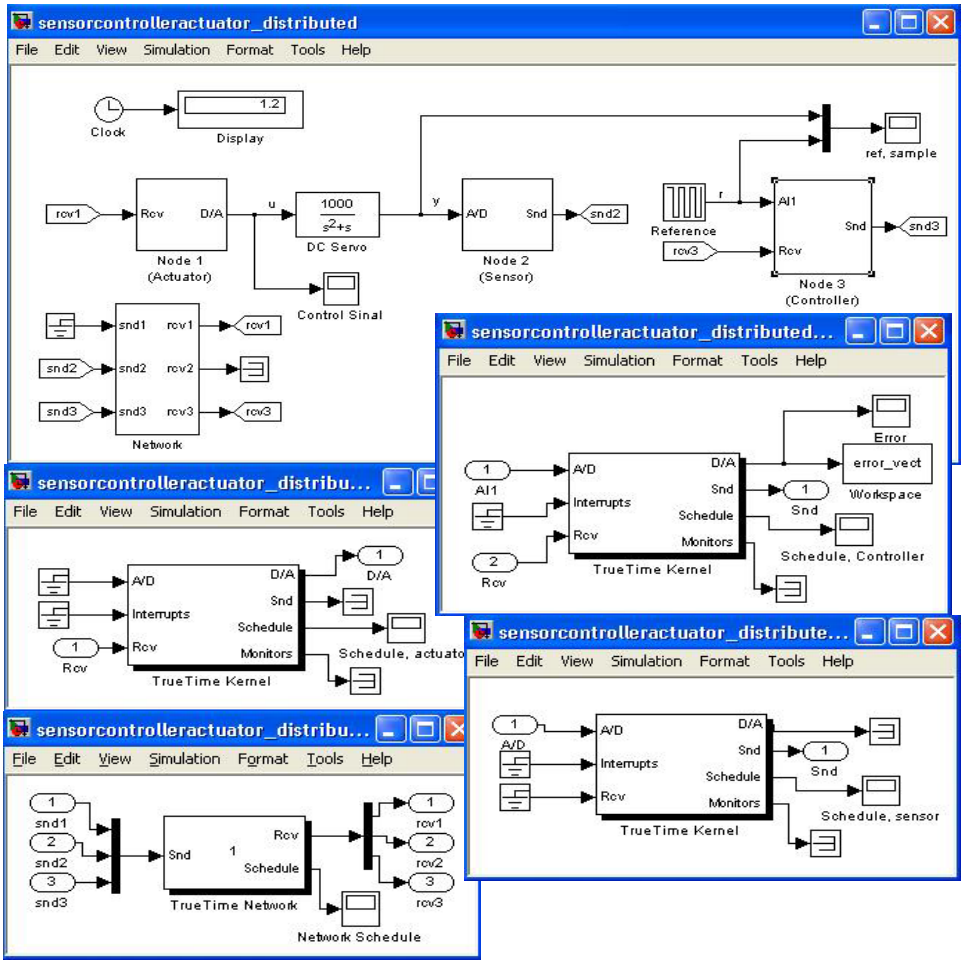

Figura 5-11 – Implementação do modelo Sensor/Controlador/Actuador usando o simulador Truetime

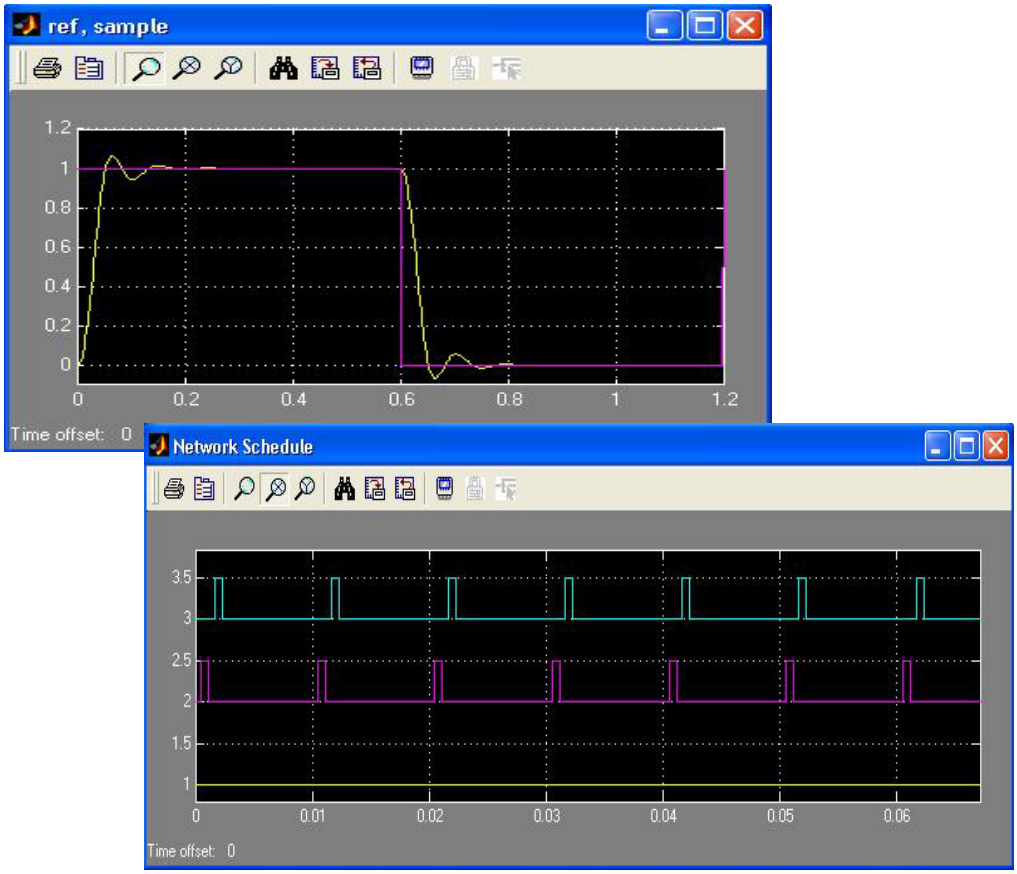

Figura 5-12 – Sinal de amostragem e sinal de referência (em cima) e escalonamento das mensagens na rede (em baixo) para o modelo totalmente distribuído usando o protocolo de comnicação *CSMA/AMP*

3.1.4. Resultados do cenário 1

Conforme esperado, a introdução de mensagens no fluxo de controlo introduz atrasos na malha de realimentação que degradam a qualidade do controlo. Para o mesmo controlador, a degradação é tanto maior quanto maior o atraso da malha. Esta observação é comprovada não apenas pela análise gráfica dos resultados mas também pelos valores do erro médio quadrático, apresentado na tabela 5-2. No entanto, para o caso apresentado, é interessante reparar que a degradação da qualidade do controlo não é muito significativa.

Modelo	Erro Médio Quadrático	Período de amostragem, h (ms)	Atraso de com. (ms)	Protocolo	*Bitrate* (kbps)
S+C+A	**0.0454**	10	--	*CSMA/AMP*	125
(S+C)/A	**0.0466**	10	0.64	*CSMA/AMP*	125
S/C/A	**0.0476**	10	1.28	*CSMA/AMP*	125

S+C+A – sensor, controlador e actuador no mesmo nó
(S+C)/A – sensor e controlador num nó e actuador num nó diferente
S/C/A – sensor, controlador e actuador em nós diferentes

Tabela 5-2 – Degradação do controlo, usando o erro médio quadrático à medida que aumenta o atraso da malha devido à introdução de mensagens

3.2. Cenário 2 – interferência (*CSMA/AMP*)

Neste cenário iremos utilizar o modelo Sensor/Controlador/Actuador, totalmente distribuído, e introduzir mais tráfego na rede, com diversas características.

Inicialmente será realizado um estudo do comportamento do sistema com pouca carga, adicionando para isso um nó de interferência a gerar tráfego aleatório. Seguidamente, aumentar-se-á a quantidade de tráfego gerado pelo nó de interferência. Por fim serão colocados dois nós a causar perturbações na rede. Numa primeira situação desta última experiência um dos nós de interferência terá prioridade mais elevada e o outro mais baixa relativamente aos nós da malha de controlo. Numa segunda situação ambos os nós de interferência possuem prioridades superiores à dos nós da malha de controlo.

3.2.1. Interferência reduzida com um nó

Nesta simulação, cujo modelo de implementação está representado na figura 5-13, as mensagens enviadas pelo nó de interferência servem unicamente para gerar tráfego na rede, com o objectivo de verificar de que forma o controlo é afectado. A interferência será gerada aleatoriamente mas ocupando, em média, uma largura de banda de 25%, com um tempo mínimo entre transmissões de 640μs que corresponde também ao tempo necessário para o envio de uma mensagem (10 *bytes*). Desta forma, o padrão de interferência poderá incluir a transmissão de várias mensagens em rajada (*burst*). Refira-se, ainda, que é utilizada uma prioridade mais elevada que a dos restantes nós que compõem a malha de controlo. A largura de banda média ocupada pelo nó de interferência (nó 1 da figura 5-13) é possível ser controlada através do

código que lhe está associado, designado por `interference_code`, e que está retratado na figura 5-14.

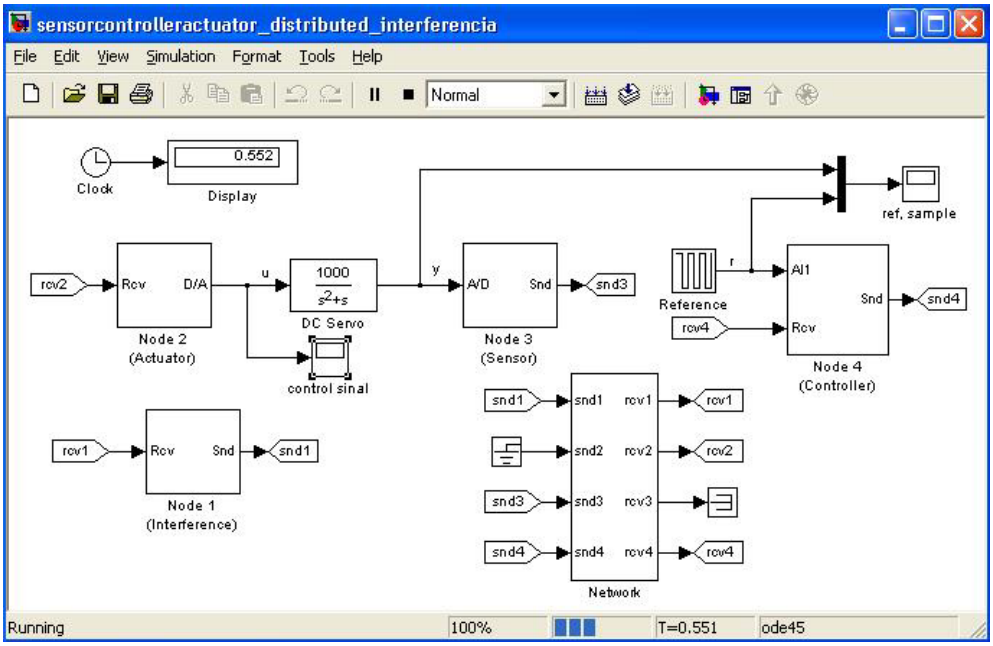

Figura 5-13 – Implementação do modelo Sensor/Controlador/Actuador com um nó de interferência usando o simulador Truetime

```
function [exectime, data] = interference_code(segment, data)

BWshare = 0.25;              % Fracção de largura de banda ocupada pelo nó
  if (rand(1) < BWshare)
    ttSendMsg(1, 1, 10, 1);  % Envia 10 bytes para o próprio nó (nó 1)
  end                        % com prioridade 1
exectime = -1;               % Terminou
```

Figura 5-14 – Listagem de código do nó de interferência

Os valores do sinal de referência e amostragem do sistema bem como o escalonamento de mensagens enviadas para a rede estão representados na figura 5-15. Na figura de cima verifica-se que o sinal de amostragem acompanha o sinal de referência com a existência de um pequeno *overshoot* nas transições do valor de referência (*set-point*). Medido o valor do erro médio quadrático (tabela 5-3), obtém-se um valor (0.0482) superior ao do modelo correspondente no cenário anterior (0.0476),

o qual se deve ao aumento do atraso no acesso ao meio por parte das mensagens de controlo, provocado pelo tráfego de interferência. Também na figura 5-15 mas em baixo pode-se ver o impacto originado pelo nó de interferência no escalonamento das mensagens na rede.

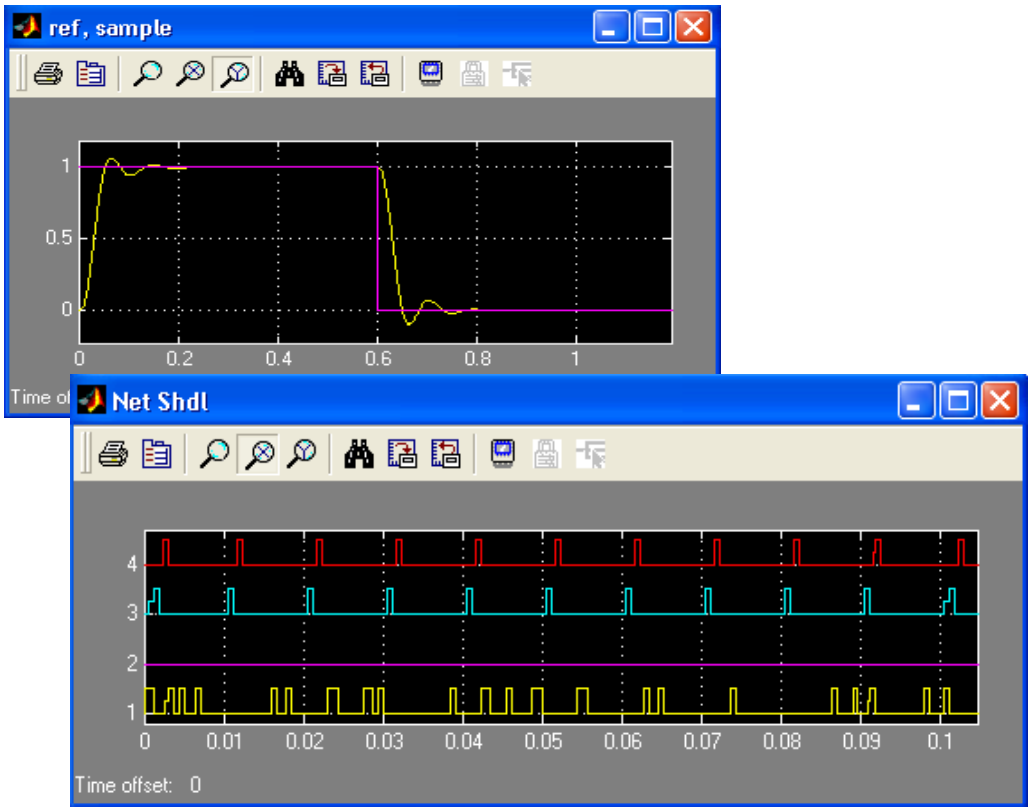

Figura 5-15 – Sinal de amostragem e sinal de referência (em cima) e tráfego de mensagens (em baixo) na rede usando o protocolo de comunicação *CSMA/AMP*

3.2.2. Interferência Elevada com um nó

A simulação seguinte, usa o mesmo modelo da anterior mas utilizando mais tráfego de interferência, com uma largura de banda média de 35% e restantes características iguais.

Os sinais obtidos por simulação podem ser visualizados na figura 5-16, não sendo visível uma degradação relativamente ao caso anterior. O valor médio do erro quadrático médio expresso na tabela 5-3 confirma essa semelhança (0.0487).

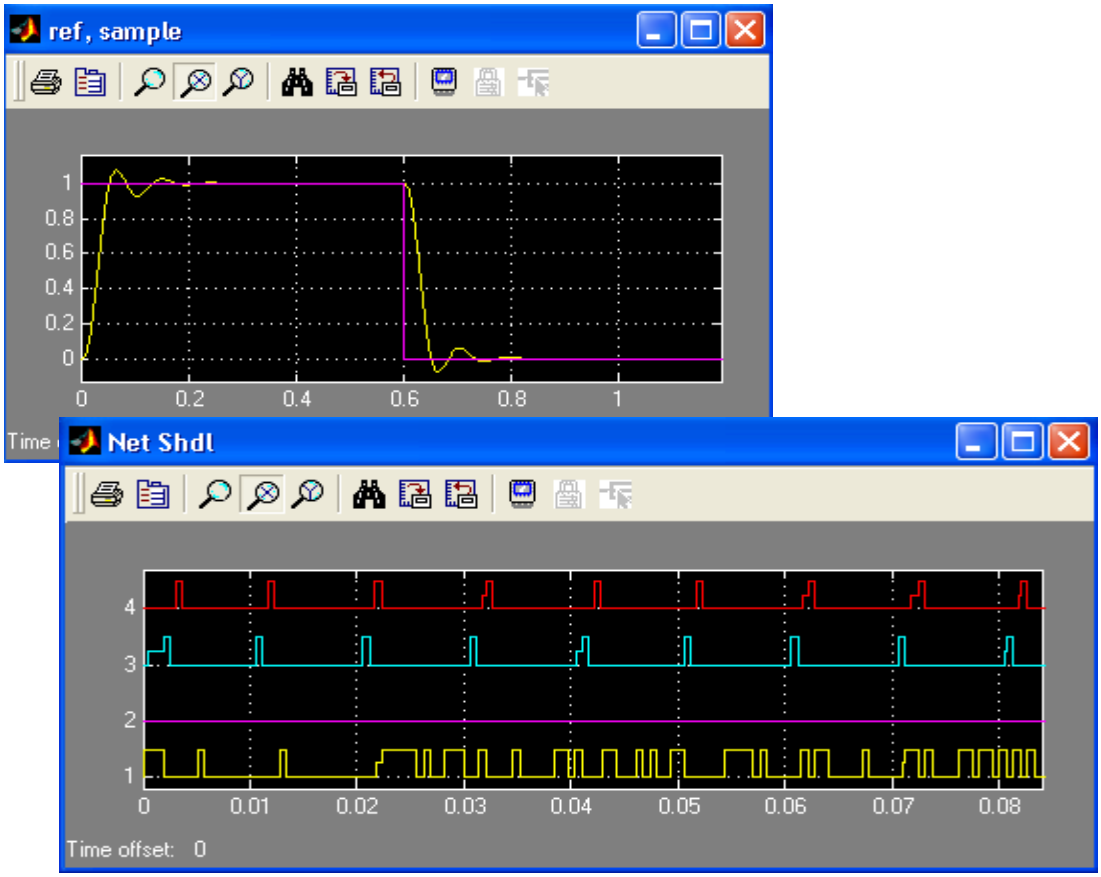

Figura 5-16 – Sinal de amostragem e sinal de referência (em cima) e tráfego de mensagens na rede (em baixo) com um nó de interferência usando o protocolo de comunicação *CSMA/AMP*

3.2.3. Interferência com dois nós

Nesta experiência introduzimos mais um nó de interferência, o que aumenta a probabilidade de *bursts*. Numa primeira situação, o segundo nó de interferência gera tráfego com prioridade mais baixa que os restantes. Assim, o impacto deste tráfego nos tempos de acesso ao meio das mensagens de controlo é apenas correspondente ao bloqueio de uma mensagem, no máximo. Numa segunda situação, utilizou-se nos dois nós de interferência prioridades mais elevadas que nos nós da malha de controlo. O modelo usado está descrito na figura 5-17.

Na primeira situação, ambos os nós de interferência ocupam uma largura de banda média de 35% e um tempo mínimo entre transmissões de 640μs. Na figura 5-18, mostram-se os sinais de controlo bem como o escalonamento de mensagens da rede.

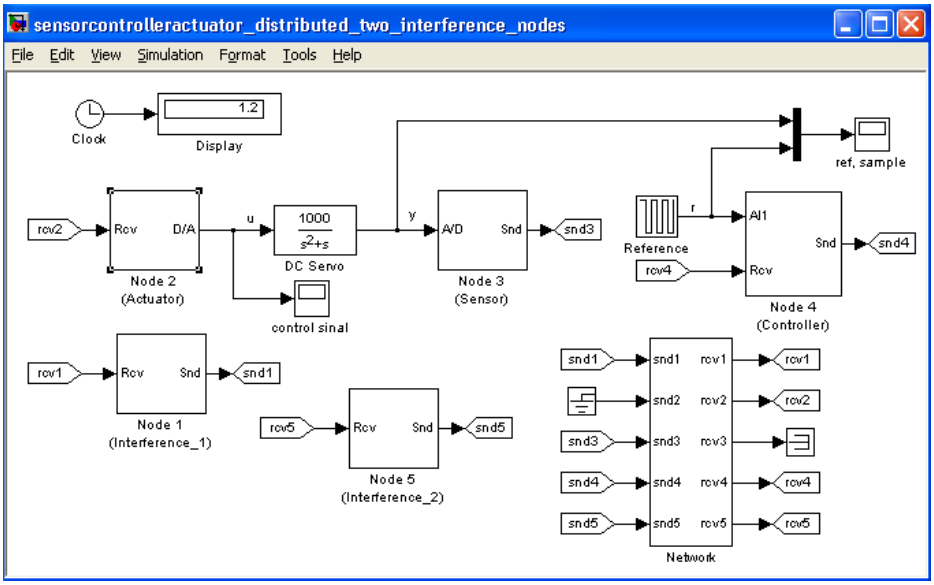

Figura 5-17 – Implementação do modelo Sensor/Controlador/Actuador com dois nós de interferência usando o simulador *Truetime*

Como se pode verificar pelo sinal de controlo (figura em cima), não existe uma degradação significativa da qualidade de controlo. O valor do erro médio quadrático obido foi de 0.0490, ligeiramente superior ao da experiência anterior. A figura mostra também o escalonamento de mensagens, onde se pode observar que o impacto do segundo nó de interferência (nó 5) é muito reduzido, razão pela qual esta situação é muito semelhante à anterior, embora a largura de banda total do tráfego de interferência tenha duplicado (cerca de 70%).

Numa última experiência colocaram-se ambos os nós de interferência a produzir tráfego de mais alta prioridade do que os nós que compõem a malha de controlo, ambos com uma largura de banda média de 30% (60% no total). Este caso permite observar o comportamento do sistema usando o protocolo *CSMA/AMP* para situações em que o tráfego de interferência é muito significativo.

Na figura 5-19 (em cima), verifica-se que a degradação da qualidade do controlo continua, ainda assim, a não ser significativa embora já seja visível um tempo de estabelecimento mais longo na transição ascendente do sinal de referência. O valor do erro médio quadrático aumentou para 0.0542.

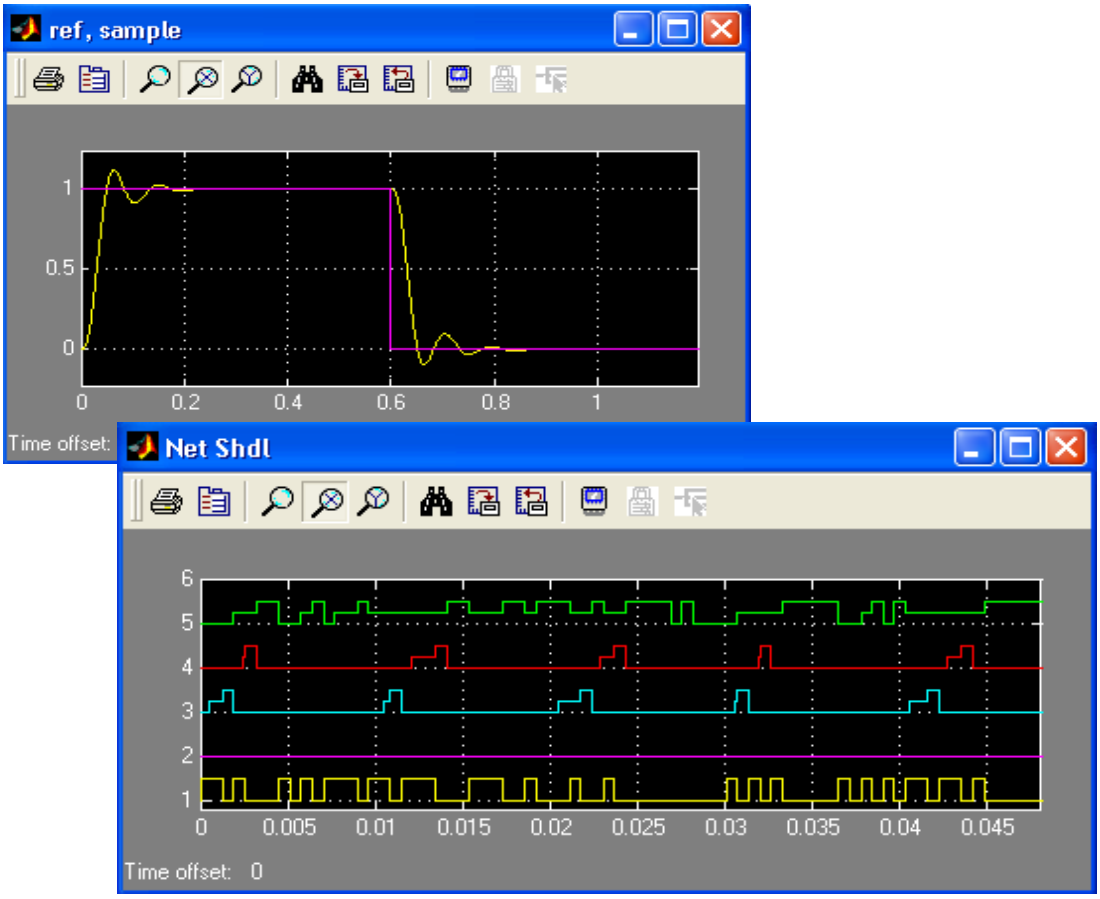

Figura 5-18 – Sinal de amostragem e de referência (em cima) e tráfego de mensagens na rede (em baixo) com dois nós de interferência usando o protocolo de comunicação *CSMA/AMP*

Para se perceber melhor a degradação maior ocorrida na transição ascendente do sinal de referência relativamente à transição descendente, é necessário verificar que mesmo com várias repetições ocorreram mais *bursts* de interferência nos instantes iniciais do que no resto da simulação. O impacto desses *bursts* é claro na figura 5-19 (em baixo), que contém uma ampliação do escalonamento da rede nos primeiros 85*ms* de uma simulação, sendo visíveis atrasos nas mensagens de controlo que chegam a ser perto do intervalo de amostragem, *h*.

3.2.4. Resultados do cenário 2

Na tabela 5-3, podemos observar os vários valores médios do erro médio quadrático para as quatro experiências realizadas nesta secção. Como o padrão de interferência é aleatório, foram realizadas vinte simulações para cada experiência e posteriormente

calculado o valor médio de todas as medições do erro médio quadrático. Estes resultados permitem-nos concluir que, ao utilizar o protocolo de acesso ao meio *CSMA/AMP*, o impacto do tráfego de interferência sobre uma malha de controlo distribuído só é significativo se a interferência tiver prioridade superior à prioridade das mensagens de controlo e se a respectiva taxa de ocupação média da rede for elevada (cerca de 60%). Caso contrário, o impacto é praticamente insignificante.

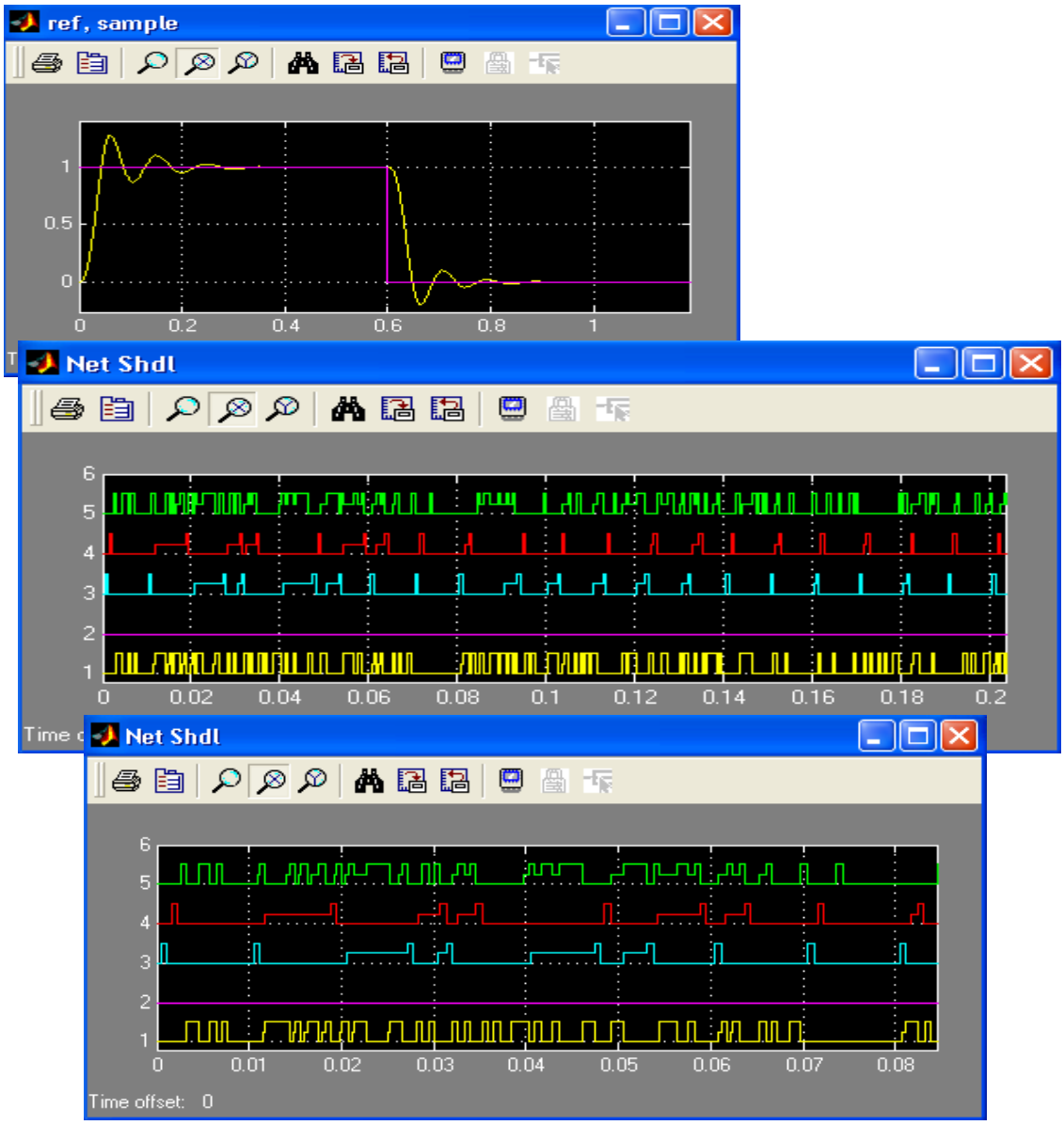

Figura 5-19 – Sinal de amostragem e de referência (em cima) e tráfego de mensagens na rede (no meio e em baixo) com dois nós de interferência de alta prioridade usando o protocolo de comunicação *CSMA/AMP*

Simulação	$\overline{MSE}$	Nó Interferência 1		Nó Interferência 2		Protocolo	*Bitrate* (Kbps)
		Prioridade	B.W.(%)	Prioridade	B.W.(%)		
1	**0.0482**	*high*	25	--	--	*CSMA/AMP*	125
2	**0.0487**	*high*	35	--	--	*CSMA/AMP*	125
3	**0.0490**	*high*	35	*low*	35	*CSMA/AMP*	125
4	**0.0542**	*high*	30	*high*+1	30	*CSMA/AMP*	125

$\overline{MSE}$ – Valor médio do Erro Médio Quadrático
B.W. – Largura de Banda (*Bandwidth*)

Tabela 5-3 – Resultados da degradação do controlo através do valor médio do erro médio quadrático em *CSMA/AMP* sob influência de perturbações na rede

3.3. Cenário 3 – *CSMA/CD*, *Token Bus*, *TDMA*

Neste cenário pretende-se avaliar o impacto do protocolo de acesso ao meio nos atrasos induzidos pela rede. Desta forma, nesta secção irão ser realizadas experiências baseadas nos cenários anteriores e utilizando os mesmos modelos. Os protocolos que se pretende comparar são, para além do *CSMA/AMP* já utilizado no cenário anterio, *CSMA/CD*, *Round Robin* e *TDMA*. Por forma a tentar comparar apenas o controlo do acesso ao meio, foram utilizados valores semelhantes aos do cenário anterior para os seguintes parâmetros: taxa de transmissão de 125kbps, tamanho total de cada mensagem de 10 *bytes*. Diga-se ainda que o período de controlo utilizado também é o mesmo, com h = 10*ms*.

3.3.1. *CSMA/CD*

O protocolo *CSMA/CD*, pelo qual se entende *Carrier Sense Multiple Access with Collision Detection*, define que estando o barramento ocupado o emissor espera até que esteja livre e então tenta transmitir. Quando uma colisão ocorrer o emissor ficará em *back off* por um tempo definido por,

$$t_{backoff} = \frac{\text{tamanho mínimo da trama}}{\text{taxa transmissão} \times R}$$

onde $R = rand(0, 2^k - 1)$ e k é o número de colisões que afectou a transmissão da mesma mensagem. Depois dessa espera, o nó irá tentar retransmitir. Contudo, depois de serem alcançadas dez colisões o intervalo para geração de valores aleatórios é mantido constante num máximo de 1023 *slots*. Ao fim de 16 tentativas o nó desiste de transmitir a trama, sendo esta descartada. Por exemplo, estando dois nós à espera

que um terceiro transmita, esses dois nós irão colidir com probabilidade 1, depois com probabilidade ½ (*k*=1), seguidamente com probabilidade ¼ (*k*=2), e assim sucessivamente

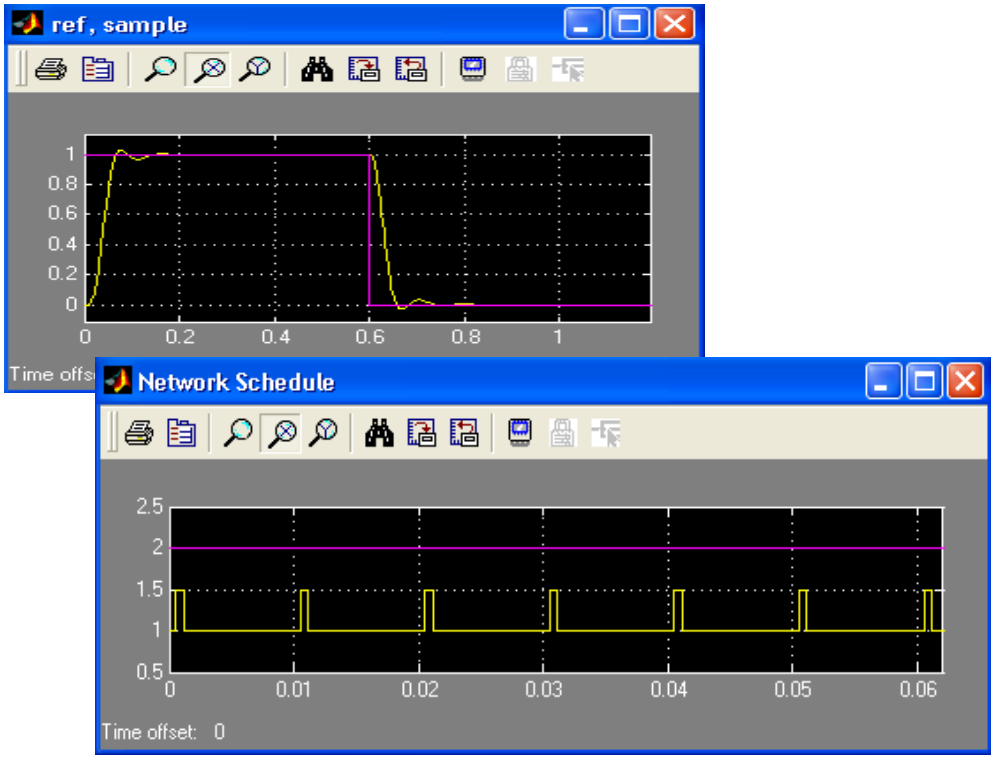

Figura 5-20 – Sinal de amostragem e de referência (em cima) e escalonamento de mensagens (em baixo) usando o *MAC CSMA/CD*

3.3.1.1. Cenário 3/1 – *CSMA/CD*

As simulações realizadas nesta primeira experiência são iguais às realizadas para o modelo (Sensor + Controlador) / Actuador na secção 3.1.2, com excepção do protocolo usado para controlo de acesso ao meio, como referido acima.

São mostrados na figura 5-20 os resultados do controlo e do escalonamento de mensagens na rede para o modelo descrito na figura 5-9. Como se pode verificar usando este protocolo e com uma rede sem tráfego, o sinal de controlo apresenta-se bastante estável. Este facto é de esperar, já que não há colisões no acesso à rede.

Também como seria de esperar, figura 5-21, os resultados obtidos para o modelo distribuído em que o sensor, o controlador e o actuador se encontram em diferentes nós, são idênticos aos obtidos no modelo anterior.

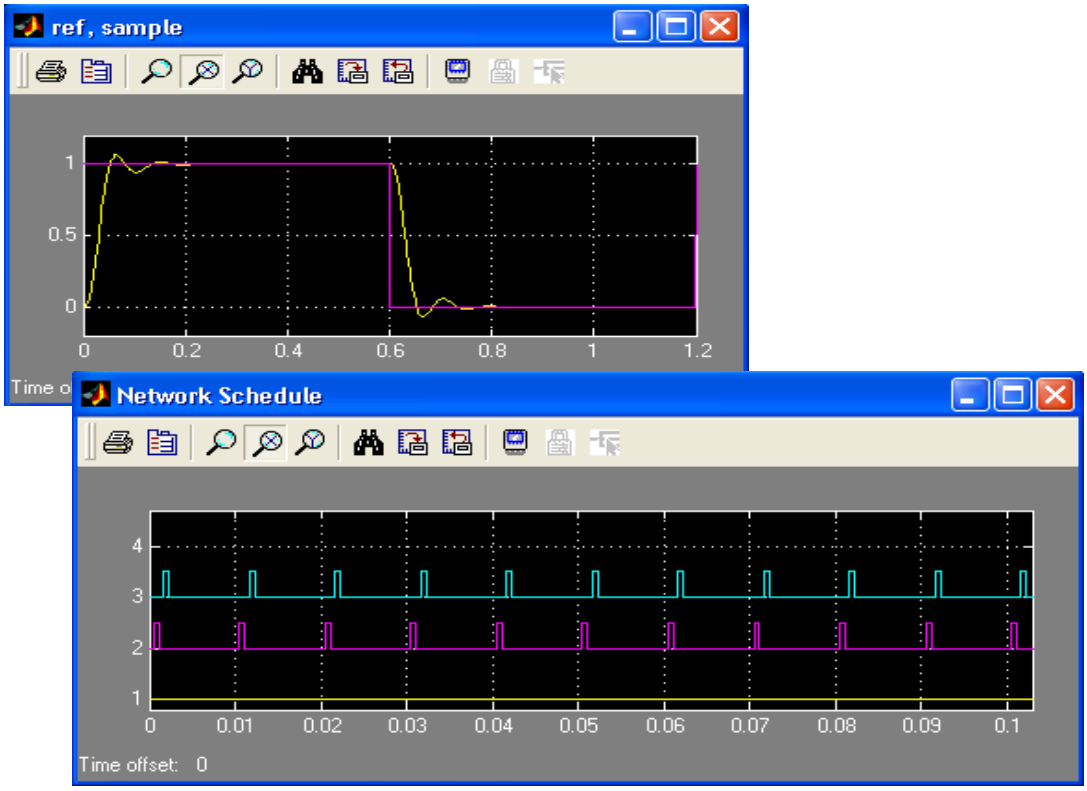

Figura 5-21 – Sinal de amostragem e referência (em cima) e escalonamento de mensagens (em baixo) usando o protocolo de comunicação *CSMA/CD*

Na tabela 5-4, podemos observar os valores médios do erro médio quadrático para as duas configurações. Como se pode verificar os valores comprovam o que foi referido anteriormente relativamente à estabilidade e desempenho do controlo.

Modelo	Erro Médio Quadrático	Período de amostragem, h (ms)	Protocolo	*Bitrate* (kbps)
(S+C)/A	**0,0466**	10	*CSMA/CD*	125
S/C/A	**0,0476**	10	*CSMA/CD*	125

(S+C)/A – sensor e controlador num nó e actuador num nó diferente

S/C/A – sensor, controlador e actuador em nós diferentes

Tabela 5-4 – Resultados da degradação do controlo utilizando *CSMA/CD*

3.3.1.2. Cenário 3/2 – *CSMA/CD*

Neste caso repetimos as simulações dos modelos apresentados em 3.2.1, 3.2.2 e 3.2.3 mas com o protocolo de acesso ao meio substituído por *CSMA/CD*.

Assim, quando se insere um nó a causar interferência (tal como na figura 5-13) o resultado em termos de sinal de controlo e escalonamento da rede é mostrado na figura 5-22. A degradação do desempenho de controlo também é mínima, com um erro quadrático médio de 0.0481. Contudo, a forma deste protocolo de acesso resolver as colisões, com espera e retransmissão, leva a alguns atrasos consideráveis das mensagens de controlo, próximos do período de amostragem.

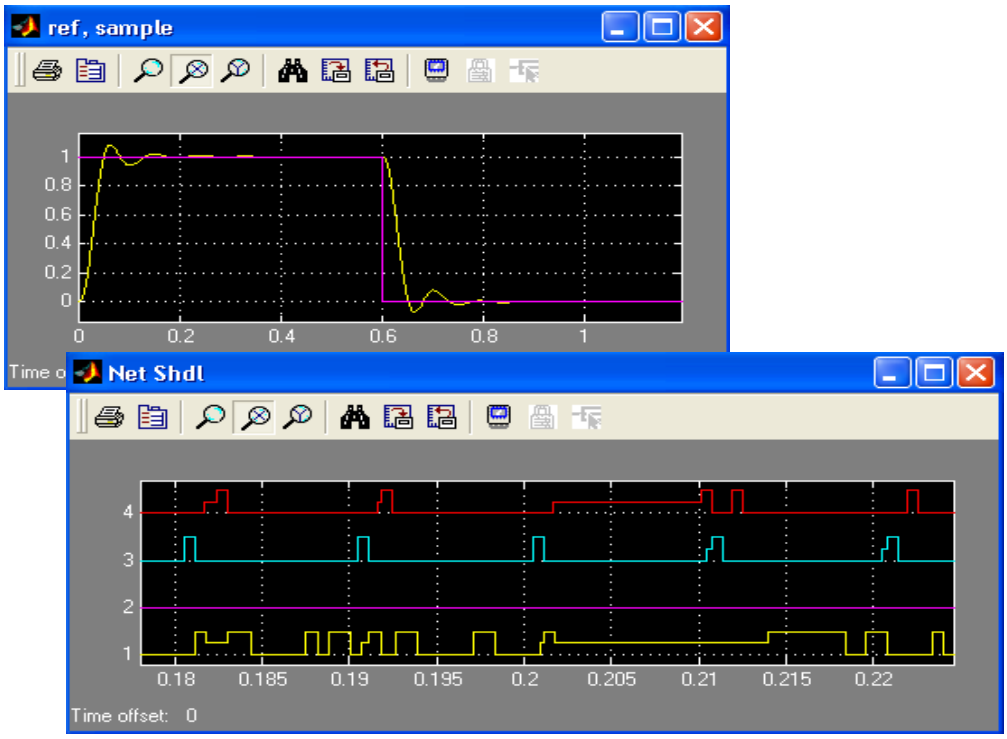

Figura 5-22 – Sinal de controlo (em cima) e tráfego de mensagens na rede (em baixo) com um nó de interferência a ocupar 25% de largura de banda usando o protocolo de comunicação *CSMA/CD*

Aumentando a largura de banda média associada ao tráfego de interferência para 35%, verificou-se que existe já uma degradação significativa do desempenho de controlo. Este facto pode ser observado na figura 5-23 e deve-se aos atrasos ainda

maiores associados à resolução das colisões. O valor do erro médio quadrático registado é já de 0.530 (tabela 5-5).

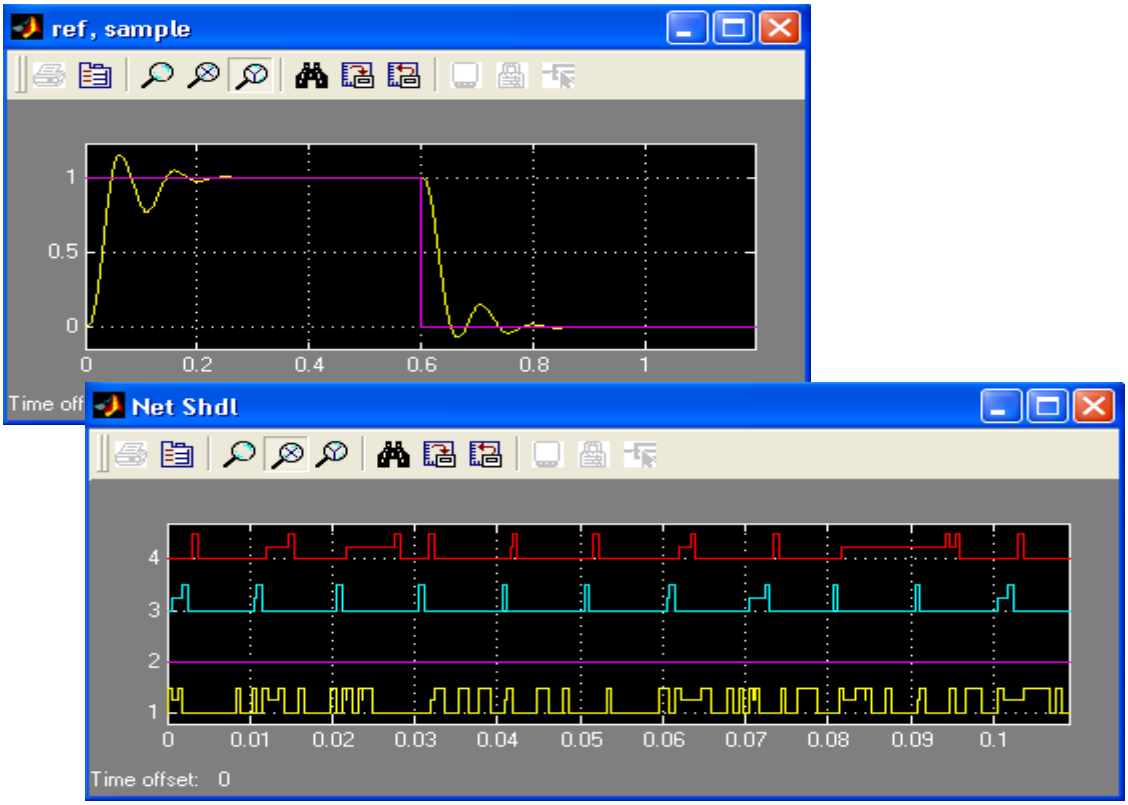

Figura 5-23 – Sinal de controlo (em cima) e tráfego de mensagens na rede (em baixo) com um nó de interferência a ocupar 35% de largura de banda usando o protocolo de comunicação *CSMA/CD*

Um aspecto de grande importância relativo ao protocolo *CSMA/CD* reside na inexistência de prioridades, estando todas as mensagens em condições semelhantes em termos de acesso ao meio. Assim, os resultados correspondentes às duas experiências com dois nós a gerar tráfego de interferência distinguem-se apenas pela largura de banda total., no primeiro caso de 70% e no segundo de 60%.

Como seria de esperar, os resultados obtidos são semelhantes em ambos os casos, apresentando-se o sistema totalmente instável (figura 5-24, em cima). Pode-se verificar pela mesma figura (em baixo) que a partir dos 250*ms* nem o sensor nem o controlador conseguem enviar qualquer mensagem. Isto deve-se ao elevado tráfego existente na rede, criado pelos processos de retransmissão associados à resolução de colisões. Este fenómeno é comummente designado por *thrashing*.

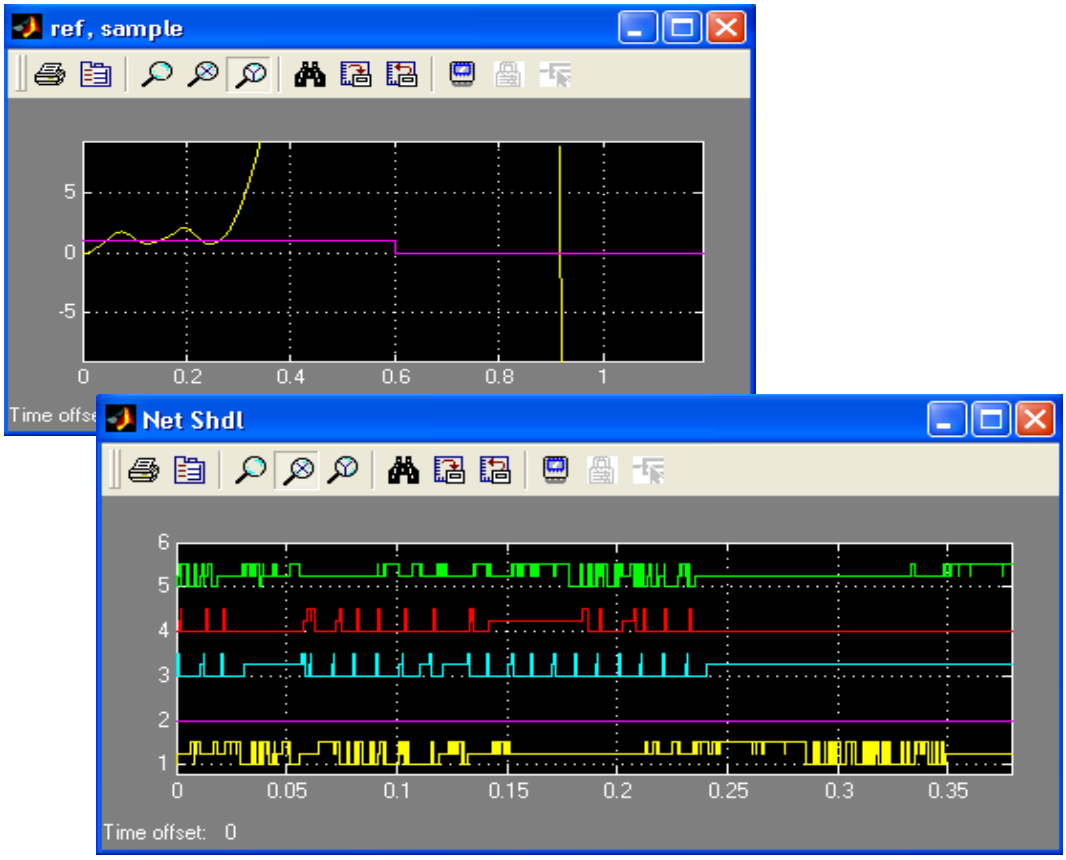

Figura 5-24 – Sinal de controlo (em cima) e tráfego de mensagens na rede (em baixo) usando o protocolo de comunicação *CSMA/CD*. Interferência de dois nós com um índice de ocupação 35% de largura de banda cada um.

Conclui-se assim que para condições de muita sobrecarga da rede, a partir aproximadamente de 60%, o protocolo de comunicação *CSMA/CD* não é adequado para aplicação em sistemas de controlo.

Na tabela 5-5 podemos observar os valores médios do erro médio quadrático obtidos para as simulações anteriormente descritas. Podemos concluir que, quando o protocolo *CSMA*/*CD* é utilizado para controlar o acesso ao meio, a presença de outro tráfego na rede é tolerável até taxas de utilização entre os 25% e os 35% (simulações 1 e 2). Para interferências com utilizações da largura de banda a partir dos 60% verifica-se o fenómeno de *thrashing* (simulações 3 e 4) o desempenho do controlo degrada-se completamente, tornando-se o sistema instável.

Simulação	$\overline{MSE}$	Nó Interferência 1		Nó Interferência 2		Protocolo	*Bitrate* (Kbps)
		Prioridade	B.W.(%)	Prioridade	3.W.(%		
1	**0.0481**	--	25	--	--	*CSMA/CD*	125
2	**0.0530**	--	35	--	--	*CSMA/CD*	125
3	**Diverge**	--	35	--	35	*CSMA/CD*	125
4	**Diverge**	--	30	--	30	*CSMA/CD*	125

$\overline{MSE}$ – Valor médio do Erro Médio Quadrático
B.W. – Largura de Banda (*Bandwidth*)

Tabela 5-5 – Qualidade do controlo com o MAC *CSMA/CD* numa rede com perturbações

3.3.2. *Round Robin* (*Token Bus*)

No protocolo *Round Robin* ou *Token Bus*, os nós apenas podem transmitir quando recebem o testemunho (*token*), o qual circula de forma fechada do nó mais prioritário para o menos prioritário. Quando na posse do *token*, cada nó pode transmitir uma única mensagem. Caso não tenham nada para transmitir passam o *token* ao nó seguinte. A passagem do *token* entre nós retira algum tempo à rede, a qual fica inacessível por um período de tempo dado por,

$$t_{idle} = \frac{\text{tamanho mínimo da trama}}{\text{taxa de transmissão}}$$

Neste caso, considerando mensagens de 10 *bytes* e velocidade de transmissão de 125Kbps, o tempo entre transmissões em que a rede fica inacessível é de 640μs.

```
function [exectime, data] = sens_ctrl_code(segment, data)

switch segment,
    case 1,
        y = ttAnalogIn(2);              % Ler entrada analogica
        r = ttAnalogIn(1);              % Ler entrada analogica: sinal de
                                        % referencia
        ttAnalogOut(1, r-y);
        P = data.K*(r-y);
        D = data.ad*data.Dold + data.bd*(data.yold-y);
        data.u = P + D;                 % saida de controlo
        data.Dold = D;
        data.yold = y;
        exectime = 0.0005;              % tempo de execuçao: 500us
    case 2,
        ttSendMsg(2, data.u, 10, 1);    % envia mensagem de 10 bytes para
                                        % nodo 2 (actuador); prioridade 1
        exectime = -1;                  % terminado
end
```

Figura 5-25 – Listagem de código do nó sensor-controlador. Tamanho da mensagem

3.3.2.1. CENÁRIO 3/1 – *TOKEN BUS*

Para o modelo (Sensor + Controlador) / Actuador obtiveram-se os resultados ilustrados na figura 5-26. Como se pode observar, o nó não transmite logo que calcula o sinal de controlo, mas sim quando lhe é atribuído o *token*. Daí o aparecimento dos tempos de espera para transmitir mesmo quando a rede está aparentemente livre. No entanto, os tempos de espera são relativamente curtos comparativamente ao período da malha de controlo pelo que o impacto no desempenho de controlo é pequeno, como se pode observar ainda na mesma figura (em baixo).

Para o modelo totalmente distribuído, os resultados da simulação são representados pela figura 5-27. As conclusões que podemos aferir são semelhantes às tiradas para o modelo anterior. Neste caso o modelo apresenta uma degradação ligeiramente superior do desempenho no controlo em relação ao modelo anterior. Isto deve-se aos atrasos superiores inerentes à existência de mais nós na rede (mais tempo para a circulação do *token*).

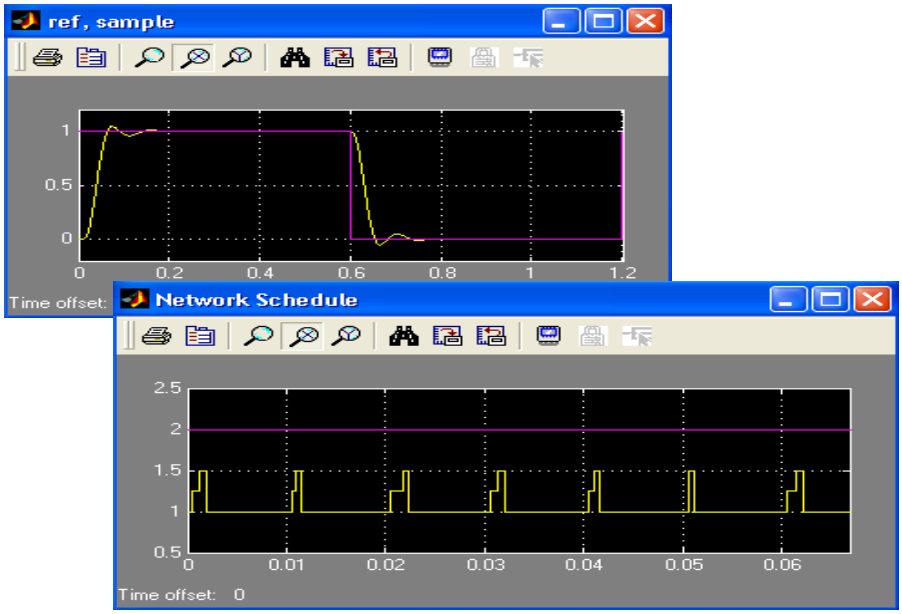

Figura 5-26 – Sinal de controlo (em cima) e escalonamento de mensagens na rede (em baixo) em *Token Bus,* no modelo (Sensor+Controlador) / Actuador.

Na tabela 5-6 podem-se ver os valores médios do erro quadrático médio obtidos para as duas simulações. Podemos observar que quando existem mais nós, logo maior atraso na circulação do *token*, a degradação do desempenho no controlo aumenta.

Figura 5-27 – Sinal de controlo (em cima) e escalonamento de mensagens (em baixo) em *Token Bus* segundo o modelo totalmente distribuído.

Para poucos nós, como é o caso, essa degradação é pequena embora seja já visível em comparação com a obtida com os protocolos *CSMA*/(*AMP* ou *CD*) em condições equivalentes, experiências 3.1.2, 3.1.3 e 3.3.1.1.

Modelo	Erro Médio Quadrático	Período de amostragem, h (ms)	Protocolo	*Bitrate* (kbps)
(S+C)/A	**0,0473**	10	*Token Bus*	125
S/C/A	**0,0489**	10	*Token Bus*	125

(S+C)/A – sensor e controlador num nó e actuador num nó diferente
S/C/A – sensor, controlador e actuador em nós diferentes

Tabela 5-6 – Erro quadrático médio, sem tráfego adicional, com *Token Bus*

3.3.2.2. Cenário 3/2 – *Token Bus*

Nesta secção mostram-se os resultados das mesmas experiências que nas subsecções 3.2.1, 3.2.2 e 3.2.3 mas usando o protocolo *Token Bus* para acesso ao meio. Os restantes parâmetros são os mesmos.

Primeiro, inseriu-se um nó de interferência a gerar tráfego com uma largura de banda média de 25%. Os resultados obtidos estão ilustrados na figura 5-28. Nota-se que,

embora o nó de interferência tenha quase permanentemente mensagens prontas para transmissão, o atraso suplementar causado no acesso ao meio pelas mensagens de controlo é limitado superiormente pela limitação na possessão do *token*. De facto, o atraso máximo de circulação do token não poderá exceder o tempo de transmissão de uma mensagem e do próprio *token* por nó, sendo, neste caso (4,48*ms*), aproximadamente metade do intervalo de amostragem, *h*. Note-se que o actuador nunca transmite qualquer mensagem, caso contrário o tempo máximo de circulação seria de 5,12*ms*.

Figura 5-28 – Sinal de controlo (em cima) e escalonamento de mensagens na rede (em baixo) em *Token Bus* com um nó de interferência a ocupar 25% de largura de banda

Na experiência seguinte, semelhante à da secção 3.2.2, aumentou-se a largura de banda média do tráfego de interferência para 35%. Foram obtidos os mesmos resultados que na experiência anterior, o que se explica pelo facto do tempo máximo de circulação do *token* ser o mesmo.

Finalmente, realizaram-se as experiências correspondentes às da secção 3.2.3, com dois nós a gerar interferência. De salientar ainda que os identificadores (IDs) dos nós de interferência são 1 e 5 (por uma questão de coerência com as experiências

anteriores). Quer isto dizer que o *token* é atribuído primeiro ao nó de interferência_1 (nó 1) depois aos nós da malha de controlo pela seguinte ordem, actuador, sensor e controlador e finalmente ao nó de interferência_2 (nó 5). A figura 5-29 apresenta os resultados obtidos para a primeira destas experiências, em que cada nó gera uma carga média de 35% da largura de banda da rede. Os resultados obtidos são, ainda assim, semelhantes aos anteriores, apenas com uma pequena degradação causada pelo aumento do tempo máximo de circulação do *token* (5,12*ms*) associado à introdução de mais um nó.

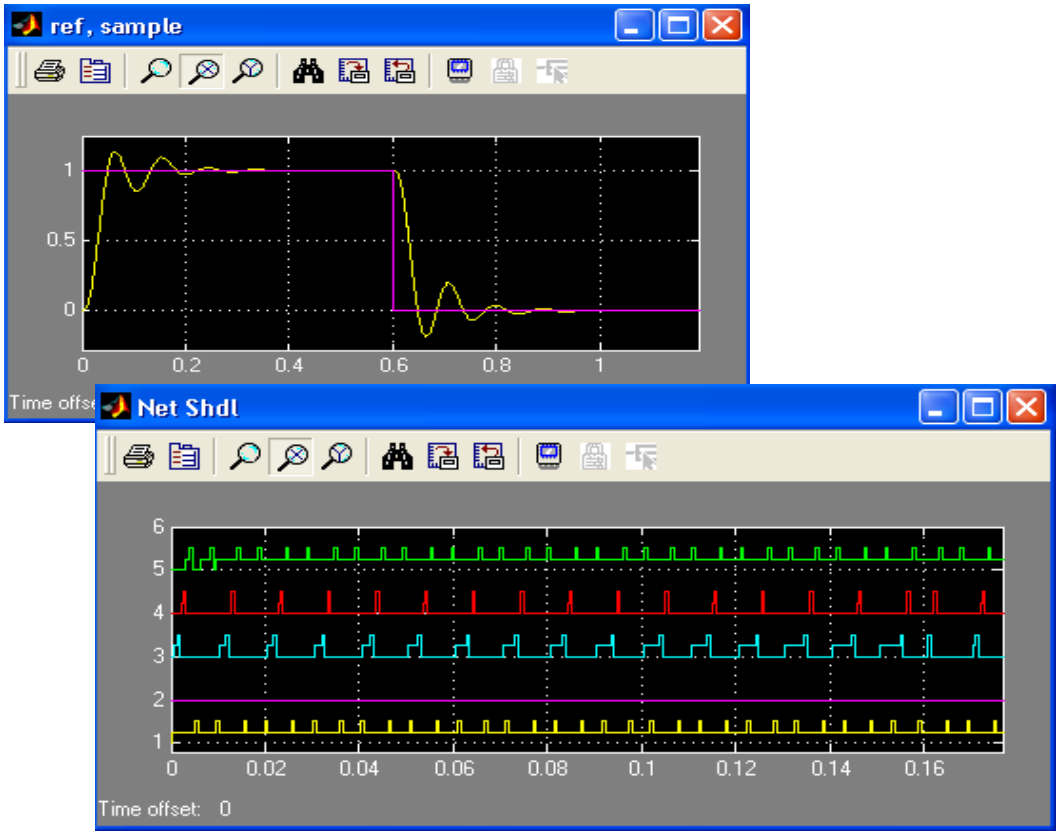

Figura 5-29 – Sinais de referência e saída (em cima) e escalonamento de mensagens na rede (em baixo) com *Token Bus* e dois nós de interferência (carga gerada de 35% da largura de banda da rede)

A figura 5-31 ilustra os resultados obtidos quando os nós de interferência geram uma carga média de 30% da largura de banda da rede. Note-se que neste caso, e porque os nós transmitem pela ordem correspondente ao seu ID, o modelo foi alterado. Assim os nós de interferência passam a ser os nós 1 e 2, o actuador o nó 3, o sensor o nó 4 e o controlador o nó 5, como mostra a figura 5-30. Os resultados são em tudo semelhantes aos da experiência anterior, conforme seria de esperar, já que o tempo de circulação máximo do *token* é o mesmo.

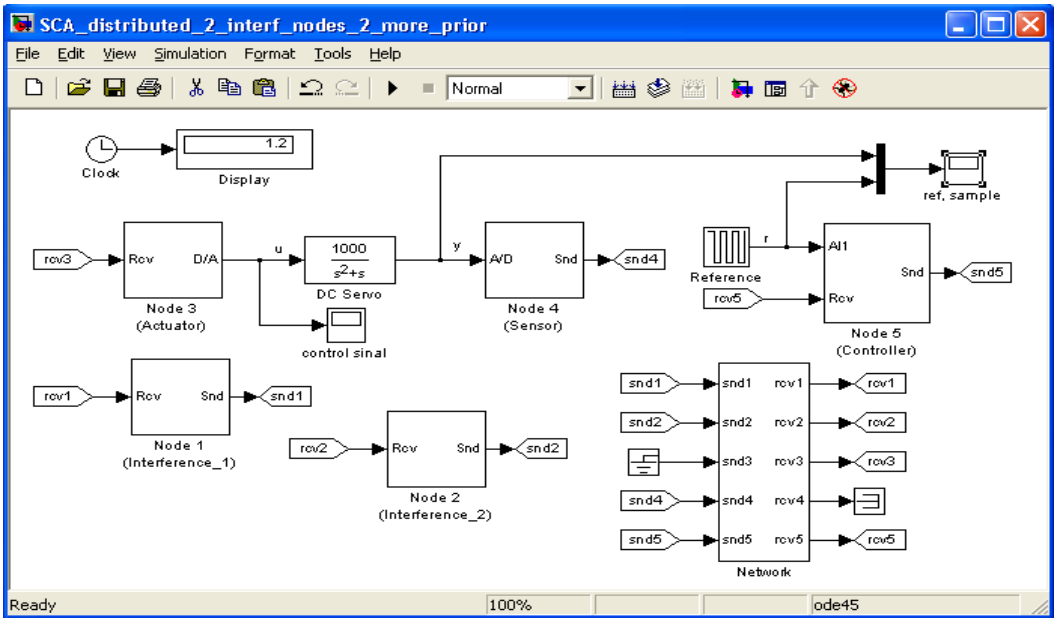

Figura 5-30 – Implementação do modelo Sensor/Controlador/Actuador com dois nós de interferência. Alteração dos IDs de forma a alterar a atribuição do *token*

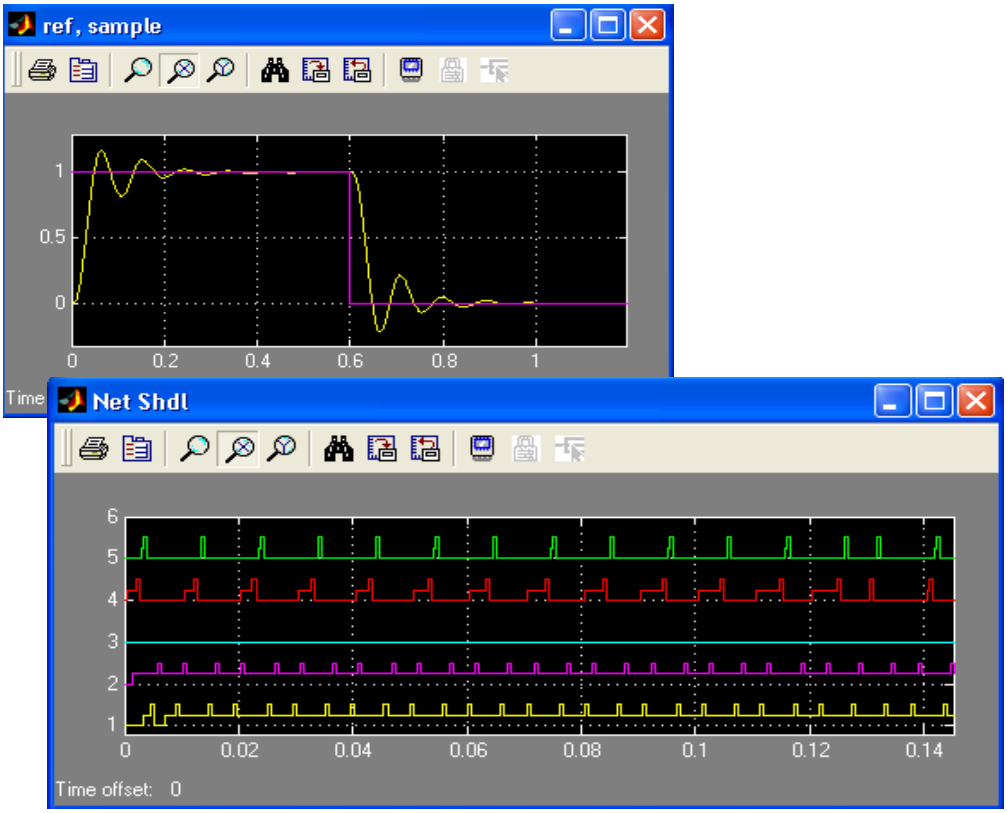

Figura 5-31 – Sinais de referência e saída (em cima) e escalonamento de mensagens na rede (em baixo) com *Token Bus* e dois nós de interferência (carga gerada de 30% da largura de banda da rede)

Também de forma semelhante à experiência anterior, podemos observar que o padrão de atraso no envio de mensagens por parte do sensor é periódico. Isto deve-se a que os instantes em que o sensor amostra o processo e tenta transmitir (que é de 10*ms* em 10*ms*) não combinarem precisamente com os instantes em que lhe é atribuído o *token*.

Uma forma de causar maior degradação no desempenho no controlo seria adicionar mais nós por forma a aumentar o atraso máximo na circulação do *token*. Outra forma seria a de mudar a ordem pela qual os nós sensor e controlador transmitem, ou seja, colocar o controlador a transmitir antes do sensor, de modo a que o *token* tenha que passar por mais nós entre as transmissões do sensor e do controlador, aumentando o atraso de controlo para quase uma volta. Neste sentido foi realizada uma outra experiência de forma a ilustrar este caso.

Figura 5-32 – Alteração da ordem de transmissão entre o controlador e o sensor. Sinais de referência e saída (em cima) e escalonamento de mensagens na rede (em baixo) com *Token Bus*.

Como se pode verificar existe quase sempre um atraso de um período de circulação do token (figura 5-32 em baixo). Este factor faz com que a degradação aumente significativamente (figura 5-32 em cima).

A tabela 5-7 mostra os resultados do erro quadrático médio obtidos nas várias experiências desta secção. Aqui podemos verificar que o impacto do tráfego de interferência é relativamente pequeno. Por outro lado, é mais significativo o número de nós e, principalmente, a ordem de circulação do token e a sua correspondência com a sequência de mensagens da malha de controlo.

Simulação	$\overline{MSE}$	Nó Interferência 1		Nó Interferência 2		Protocolo	*Bitrate* (Kbps)
		Sequência	B.W.(%)	Sequência	B.W.(%)		
1	**0.0505**	I_1-A-S-C	25	--	--	*Token Bus*	125
2	**0.0508**	I_1-A-S-C	35	--	--	*Token Bus*	125
3	**0.0512**	I_1-A-S-C-I_2	35	I_1-A-S-C-I_2	35	*Token Bus*	125
4	**0.0529**	I_1-I_2-A-S-C	30	I_1-I_2-A-S-C	30	*Token Bus*	125
5	**0,0794**	I_1-I_2-A-C-S	30	I_1-I_2-A-C-S	30	*Token Bus*	125

$\overline{MSE}$ – Valor médio do Erro Médio Quadrático
B.W. – Largura de Banda (*Bandwidth*)
I_1 – Nó Interferência 1; I_2 – Nó de Interferência 2; A – Actuador; S – Sensor; C – Controlador

Tabela 5-7 – Valor médio do erro quadrático médio para o protocolo *Round Robin* (*Token Bus*) numa rede com perturbações.

3.3.3. *TDMA*

De acordo com este protocolo de acesso ao meio, cada nó possui um *time-slot* onde tem acesso exclusivo ao *bus* e que está pré-definida no tempo. Se no seu *time-slot* não conseguir transmitir a trama na totalidade, a transmissão continua no seu *time-slot* seguinte. A duração do *time-slot* é determinada com base na expressão:

$$t_{slot} = \frac{\text{slot size}}{\text{data rate}}$$

Os parâmetros, *slot size* e *data rate* são definidos na janela de parâmetros do bloco de rede, descrita pela figura 5-4. No nosso caso, por compatibilidade com as experiências anteriores, definimos o *slot size* igual a 10 *bytes*.

3.3.3.1. CENÁRIO 3/1 – *TDMA*

Neste cenário foram repetidas as experiências efectuadas nas secções 3.1.2 e 3.1.3, com os mesmos modelos mas utilizando o protocolo *TDMA* para acesso ao meio.

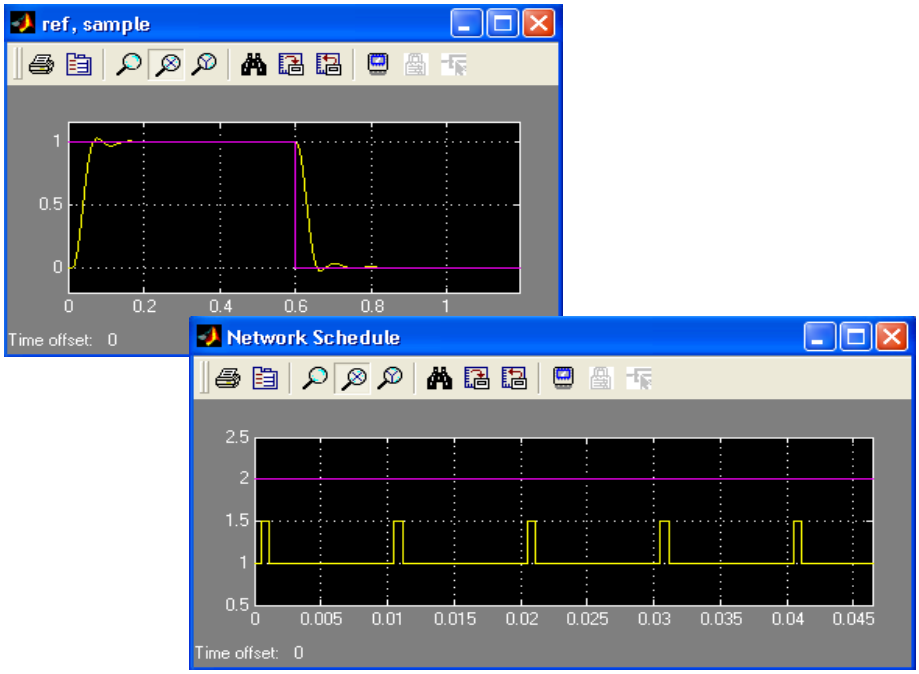

Figura 5-33 – Sinais de referência e saída (em cima) e escalonamento da rede em *TDMA* (em baixo) para o modelo Sensor+Controlador/ Actuador

Na primeira experiência o protocolo *TDMA* foi configurado para existir unicamente um nó a transmitir, sensor+controlador, pelo que fica com acesso exclusivo à rede (todos os *time-slots* são-lhe atribuídos). Conforme se pode ver pela figura 5-33 não se nota o eventual fraccionamento das mensagens já que o nó transmite em *time-slots* consecutivos. Também não se verifica nenhum atraso adicional. Assim, o desempenho da malha de controlo é semelhante ao dos protocolos *CSMA*/(*CD* e *AMP*) nas mesmas condições.

Na configuração do modelo totalmente distribuído, com 3 nós, o protocolo *TDMA* foi configurado de modo a alternar a atribuição de *time-slots* entre os dois únicos nós que enviam mensagens (sensor e controlador). A figura 5-34 mostra os resultados obtidos, onde é visível o fraccionamento das mensagens que ocorre devido aos assíncronismo entre o início dos *time-slots* e as transmissões. Como as mensagens não são enviadas no mesmo *time-slot*, esta situação vai originar um atraso na aplicação do sinal de controlo sobre o processo, conduzindo assim a uma degradação do controlo. Esse facto pode ser observado pelos valores do erro quadrático médio da tabela 5-8.

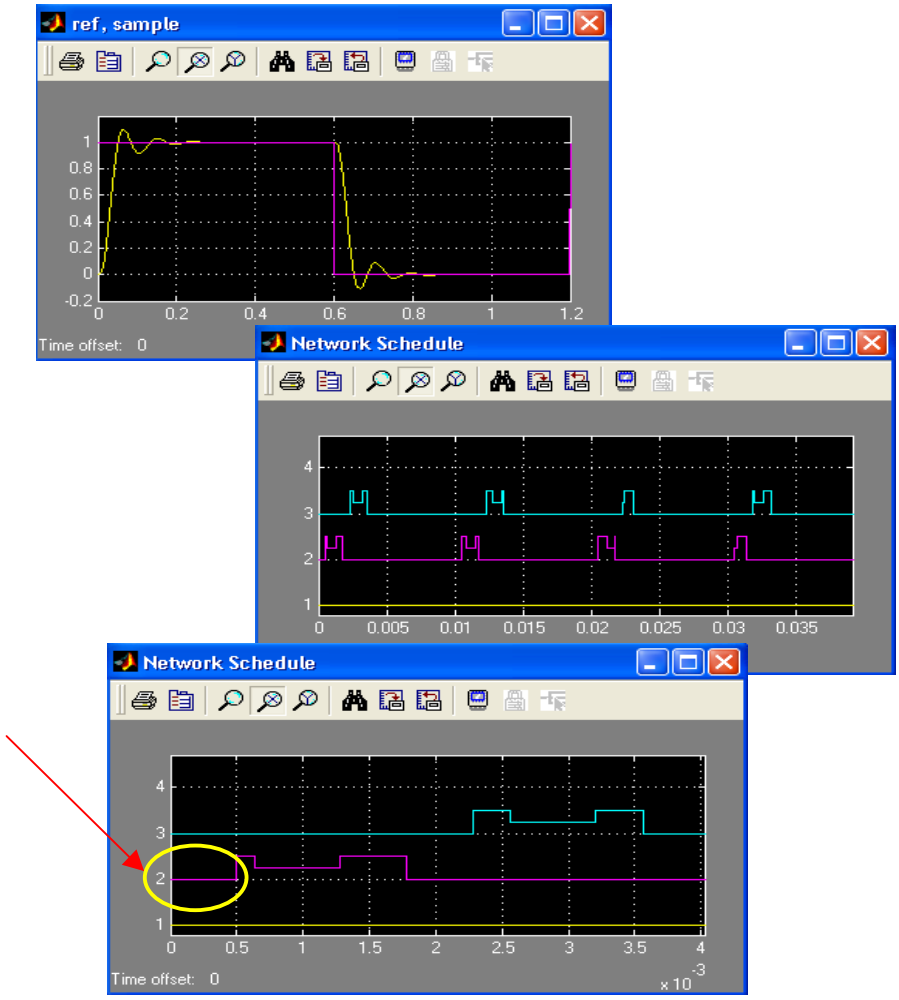

Figura 5-34 – Sinais de referência e saída (em cima) e escalonamento da rede com *TDMA* (no meio) para o modelo totalmente distribuído. Tempo de amostragem do processo pelo sensor (em baixo)

```
function [exectime, data] = sens_code(segment, data)

switch segment,
  case 1,
      data.y = ttAnalogIn(1);       % Lê valor analógico do processo
      exectime = 0.0005;            % Tempo de execução 500us
  case 2,
      ttSendMsg(4, data.y, 10, 2);% Envia mensagem de 10 bytes para nó 4
                                    %(controlador), prio 2
      exectime = -1;                % Terminou
end
```

Figura 5-35 – Tempo de execução da amostragem do processo pelo sensor

A figura 5-34 mostra ainda (em baixo), o tempo que o sensor demora a realizar a amostragem do processo (500μs, ver figura 5-35) antes de enviar a respectiva mensagem e que, nessa instância, causa um desperdício de parte do *time-slot*. A transmissão da mensagem do sensor ainda se inicia nesse *time-slot* mas já só termina no *time-slot* seguinte.

Modelo	Erro Médio Quadrático	Período de amostragem, h (ms)	Protocolo	*Bitrate* (kbps)
(S+C)/A	**0,0466**	10	*TDMA*	125
S/C/A	**0,0497**	10	*TDMA*	125

(S+C)/A – sensor e controlador num nó e actuador num nó diferente

S/C/A – sensor, controlador e actuador em nós diferentes

Tabela 5-8 – Erro quadrático médio em *TDMA* para os cenários 3/1

Por sua vez, o controlador só inicia um novo ciclo quando recebe a mensagem completa, demorando também 500μs para executar o algoritmo de controlo antes de iniciar a transmissão da mensagem de actuação.

De acordo com a configuração efectuada, nestas circunstâncias irá sempre ocorrer fraccionamento das mensagens entre *time-slots* consecutivos do mesmo nó, causando um atraso na malha de controlo. Este atraso é, contudo, relativamente pequeno, isto é, de 2 *time-slots* adicionais (1,28ms).

3.3.3.2. Cenário 3/2 – *TDMA*

À semelhança das secções 3.2.1, 3.2.2 e 3.2.3, esta secção analisa o comportamento de uma malha distribuída de controlo quando se insere tráfego adicional na rede, usando o protocolo *TDMA* para controlo de acesso ao meio.

Numa primeira experiência adicionámos ao sistema, na configuração totalmente distribuída, um nó de interferência a gerar uma carga média de 25% da largura de banda da rede. Os resultados obtidos estão ilustrados na figura 5-36 e mostram que o controlo não sofreu degradação relativamente à situação sem interferência.

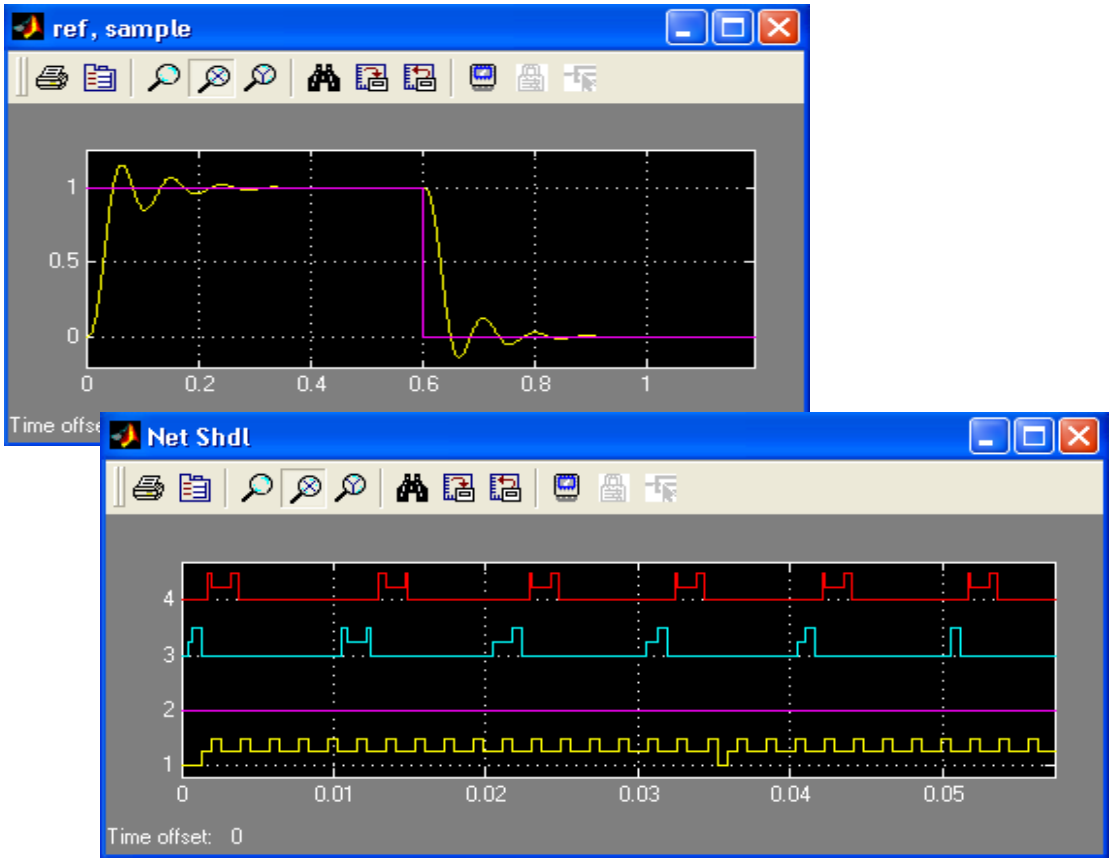

Figura 5-36 – Sinais de referência e saída (em cima) e escalonamento da rede (em baixo) com *TDMA* e um nó a gerar interferência de 25% de largura de banda.

Na figura 5-37 é mostrado, com mais pormenor, o escalonamento das mensagens apresentado na figura 5-36, mostrando o impacto do nó de interferência que se reflecte apenas num aumento do tempo entre *time-slots* consecutivas de cada nó, que agora é de 2 *time-slots* em vez de 1, como era na situação precedente. Assim, quando há fraccionamento das mensagens, cada mensagem demora mais 1,28*ms* a ser entregue ao destino, sendo este atraso a única diferença relativamente ao cenário precedente 3/1, totalmente distribuído.

Seguidamente, realizou-se uma segunda simulação para medir o impacto no desempenho do controlo quando o nó de interferência ocupa uma largura de banda mais elevada que neste caso será de 35%. Como seria de esperar, o desempenho do controlo não sofreu qualquer alteração porque cada um dos nós transmite em instantes bem definidos, independentemente das transmissões dos outros nós. Assim, qualquer que fosse o valor de carga gerada pelo nó de interferência, nada se iria alterar em relação ao desempenho do controlo.

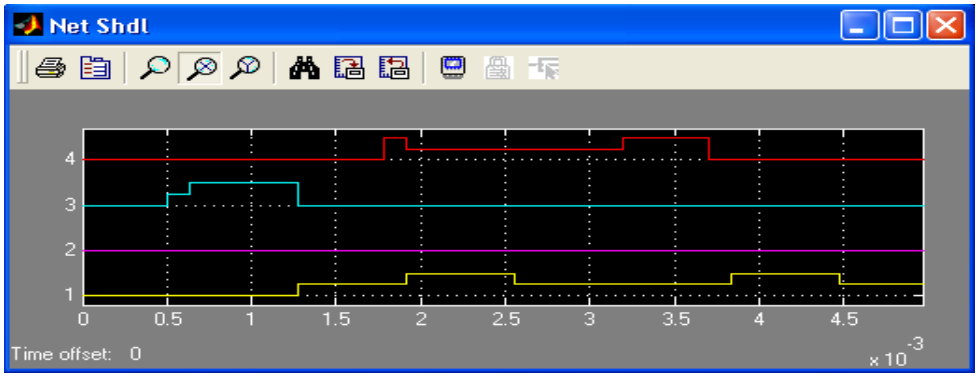

Figura 5-37 – Atribuição de *time-slots*; tempos de processamento de tarefas no interior de cada nó; envio fraccionado das mensagens em *TDMA*

Finalmente, utilizou-se mais outro nó a causar interferência na rede (nó 5) e obtiveram-se os resultados apresentados na figura 5-38. Como se pode verificar, nesta situação

Figura 5-38 – Sinais de referência e saída (em cima) e escalonamento de mensagens (em baixo) com *TDMA* e dois nós de interferência.

o fraccionamento das mensagens leva a atrasos maiores causados por mais um *time-slot* no ciclo *TDMA*. Este *time-slot* adicional pode afectar a transmissão de ambas as

mensagens de controlo causando mais um aumento de 1,28*ms* no atraso da malha de controlo relativamente à situação precedente. Este atraso vai degradar, embora de uma forma pouco significativa, a qualidade do controlo.

Tal como nas experiências correspondentes com os protocolos anteriores, também se utilizou uma carga gerada pelos nós de interferência de 30% da largura de banda da rede e com diferente ordem de transmissão no ciclo *TDMA* (1→5→3→4 em vez de 1→3→4→5 como anteriormente). Os resultados obtidos em simulação são semelhantes aos anteriores, com uma ligeira degradação do desempenho do controlo no primeiro caso e uma melhoria no segundo (ver tabela 5-9)

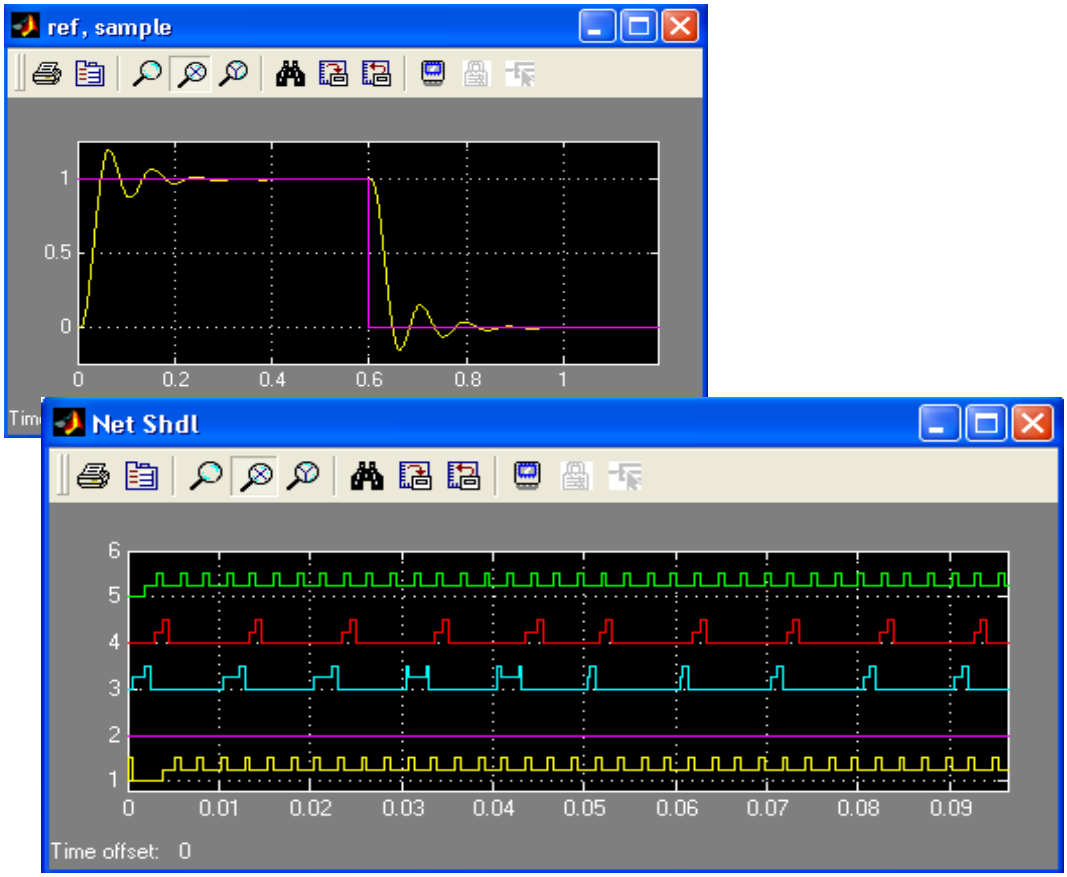

Figura 5-39 – Alteração da ordem de transmissão entre o controlador e o sensor. Sinais de referência e saída (em cima) e escalonamento de mensagens na rede (em baixo) com *TDMA*

A diferença verificada nas duas últimas experiências indica um impacto da ordem dos *time-slots* no ciclo *TDMA* sobre o atraso da malha de controlo. Como se pode verificar pela figura 5-39 (em baixo) e comparativamente à situação anterior (figura 5-38 em baixo) pode-se observar que a mudança de ordem do ciclo *TDMA* permitiu obter uma

melhor correspondência entre os instantes de transmissão dos nós de controlo e as respectivas *time-slots*. Nomeadamente, o tempo de execução do controlador é agora frequentemente sobreposto com as time-slots dos nós 1 e 5, permitindo que o atraso de controlo seja inferior a um ciclo TDMA, sendo menor que no caso anterior implicando assim uma menor degradação no controlo. Através de uma análise mais pormenorizada, para as situações anteriores em que as ordens de transmissão no ciclo *TDMA* são 1→5→3→4 e 1→5→4→3 (mudança de ordem de transmissão do controlador e sensor), podemos referir que para o primeiro caso o tempo de atraso entre amostragem-actuação está compreendido no intervalo [4,84; 6,12]*ms* e para o segundo caso entre [3,06; 5,12]*ms*. Estes valores explicam bem a razão pela qual a degradação do desempenho no controlo é menor.

Podemos referir ainda que para as quatro primeiras situações analisadas com este protocolo, tal como no caso de *Token-Bus*, não existe uma degradação significativa do desempenho do controlo com a introdução de tráfego de interferência. A degradação verificada está associada ao aumento do nós e consequente aumento do ciclo *TDMA*. Este facto está patente nos valores médios do erro quadrático médio apresentados na tabela 5-9.

Simulação	$\overline{MSE}$	Nó Interferência 1		Nó Interferência 2		Protocolo	*Bitrate* (Kbps)
		Ciclo	B.W.(%)	Ciclo	B.W.(%)		
1	**0.0513**	*1-3-4*	25	--	--	*TDMA*	125
2	**0.0513**	*1-3-4*	35	--	--	*TDMA*	125
3	**0.0543**	*1-3-4-5*	35	*1-3-4-5*	35	*TDMA*	125
4	**0.0550**	*1-5-3-4*	30	*1-5-3-4*	30	*TDMA*	125
5	**0.0513**	*1-5-4-3*	30	*1-5-4-3*	30	*TDMA*	125

$\overline{MSE}$ – Valor médio do Erro Médio Quadrático
B.W. – Largura de Banda (*Bandwidth*)

Tabela 5-9 – Resultados da degradação do controlo através do valor médio do erro médio quadrático em *TDMA*

De uma forma geral, um factor que parece condicionar o desempenho do controlo é o ajuste dos instantes em que cada nó transmite aos instantes em que decorrem os respectivos *time-slots* no ciclo *TDMA*. O ideal seria que os nós de controlo transmitissem nos instantes em que os respectivos *time-slots* se iniciam e que estes tivessem a duração adequada para evitar fraccionamento de mensagens mas isto nem sempre é possível de conseguir devido às variáveis em jogo (tempos de execução das tarefas, tempos de transmissão das mensagens e período de controlo)

4. CONCLUSÃO

Grande parte dos sistemas de controlo são sistemas distribuídos, constituídos por nós ligados por uma rede de comunicação pela qual comunicam. A utilização de serviços de comunicação para transferir informação de controlo aumenta os atrasos dessa informação, que poderão ser mais ou menos determinísticos, dependendo do protocolo de comunicação usado. Este indeterminismo pode ser reduzido através de uma escolha apropriada de técnicas e plataformas de implementação, tais como modelos de disparo das mensagens e protocolos de acesso ao meio. Para estudar especificamente o impacto dos protocolos de acesso ao meio foi usada uma ferramenta de simulação, o *TrueTime*, que facilita a simulação de modelos simples de sistemas de controlo distribuídos usando diferentes redes.

Nas simulações efectuadas utilizaram-se situações o mais idênticas possível à excepção dos próprios protocolos. Nomeadamente, utilizou-se um processo correspondente a um servo mecanismo (controlo de posição), um período de controlo de 10*ms*, um taxa de transmissão da rede de 125Kbps e mensagens com tamanho total de 10 *bytes* incluindo dados e controlo.

Depois criaram-se dois cenários principais baseados no protocolo *CSMA/AMP* e ainda um terceiro cenário em que se repetiam as experiências dos dois cenários anteriores mas usando os protocolos *CSMA/CD*, *Token-Bus* e *TDMA*. No primeiro cenário verificámos o impacto da distribuição numa malha de controlo, usando 3 configurações que usam zero, uma ou duas mensagens na malha de realimentação e sem mais nenhum tráfego adicional. Verificou-se que à medida que se iam dispersando por vários nós as funcionalidades básicas de um sistema de controlo em malha fechada, isto é, sensor, controlador e actuador, com o consequente aumento do número de mensagens na malha de realimentação, ia-se notando uma degradação crescente do desempenho de controlo.

A tabela 5-10 mostra de forma condensada os resultados das simulações realizadas nos cenários 1 e 3/1. Comparativamente entre os vários protocolos, note-se o bom desempenho dos protocolos assíncronos, *CSMA*/(*AMP* e *CD*), que transmitem assim que o *bus* está livre. Note-se que nestes cenários não há colisões no acesso ao meio. Por seu lado, os protocolos com controlo de transmissão, *Token-Bus* e *TDMA*, geram uma degradação adicional causada pela falta de sincronismo entre os instantes em que os nós pretendem transmitir e os instantes em que o protocolo de acesso ao meio lhes permite efectivamente transmitir. Assim, surgem alguns atrasos adicionais que levam a uma ligeira degradação do controlo.

Modelo	Erro Quadrático Médio	Atraso de comunicação (*ms*)	Protocolo
S+C+A	**0.0454**	--	*CSMA/AMP*
(S+C)/A	**0.0466**	0.64	*CSMA/AMP*
S/C/A	**0.0476**	1.28	*CSMA/AMP*
(S+C)/A	**0,0466**	0.64	*CSMA/CD*
S/C/A	**0,0476**	1.28	*CSMA/CD*
(S+C)/A	**0,0473**	1.92	*Token-Bus*
S/C/A	**0,0489**	3.2	*Token-Bus*
(S+C)/A	**0,0466**	0.64	*TDMA*
S/C/A	**0,0497**	1.28	*TDMA*

S+C+A – sensor, controlador e actuador no mesmo nó
(S+C)/A – sensor e controlador num nó e actuador num nó diferente
S/C/A – sensor, controlador e actuador em nós diferentes

Tabela 5-10 – Comparação do erro quadrático médio nos cenários 1 e 3/1 para os diferentes protocolos de acesso ao meio

No segundo cenário analisou-se o impacto da presença de tráfego adicional na rede, gerando interferência sobre o fluxo de informação de controlo. Neste caso utilizou-se a configuração totalmente distribuída para o sistema de controlo, com três nós, e realizaram-se várias experiências, primeiro com um nó de interferência e dois níveis de carga gerada, e depois com dois nós de interferência e diferentes prioridades relativas ou diferentes ordenações do ciclo de transmissão.

A tabela 5-11 mostra de forma condensada os resultados obtidos nos cenários 2 e 3/2. Os melhores resultados continuam a ser para *CSMA/AMP*, embora haja um impacto crescente e directo do tráfego adicional. Para este protocolo, verificou-se também que quando existe uma forte perturbação na rede com tráfego de mais alta prioridade, a degradação do controlo é significativa mas, por outro lado, se o tráfego da malha de controlo possuir prioridade mais alta o controlo não sofre degradação significativa.

Usando o protocolo de acesso ao meio *CSMA/CD* verifica-se que como este não estabelece prioridades na comunicação, o desempenho no controlo é afectado de acordo com a carga na rede. Para valores de ocupação da rede acima de um determinado valor (em torno dos 50%), verifica-se um fenómeno de *thrashing*, em que a capacidade de comunicação efectiva da rede diminui substancialmente, levando a atrasos muito grandes e instabilidade no sistema de controlo. Por esta razão este protocolo não é muito adequado para aplicação em sistemas de controlo distribuídos em que a possibilidade de ocorrência de sobrecarga na rede exista.

Simulação	$\overline{MSE}$	Nó Interferência 1		Nó Interferência 2		Protocolo
		Prioridade	B.W.(%)	Prioridade	B.W.(%)	
1	**0.0482**	*high*	25	--	--	*CSMA/AMP*
2	**0.0487**	*high*	35	--	--	*CSMA/AMP*
3	**0.0490**	*high*	35	*low*	35	*CSMA/AMP*
4	**0.0542**	*high*	30	*high*+1	30	*CSMA/AMP*
		Prioridade	B.W.(%)	Prioridade	B.W.(%)	
1	**0.0481**	--	25	--	--	*CSMA/CD*
2	**0.0530**	--	35	--	--	*CSMA/CD*
3	**Diverge**	--	35	--	35	*CSMA/CD*
4	**Diverge**	--	30	--	30	*CSMA/CD*
		Sequência	B.W.(%)	Sequência	B.W.(%)	
1	**0.0505**	I_1-A-S-C	25	I_1-A-S-C-I_2	--	*Token Bus*
2	**0.0508**	I_1-A-S-C	35	I_1-I_2-A-S-C	--	*Token Bus*
3	**0.0512**	I_1-A-S-C-I_2	35	I_1-I_2-A-C-S	35	*Token Bus*
4	**0.0529**	I_1-I_2-A-S-C	30	I_1-A-S-C-I_2	30	*Token Bus*
5	**0,0794**	I_1-I_2-A-C-S	30	I_1-I_2-A-S-C	30	*Token Bus*
		Ciclo	B.W.(%)	Ciclo	B.W.(%)	
1	**0.0513**	*1-3-4*	25	--	--	*TDMA*
2	**0.0513**	*1-3-4*	35	--	--	*TDMA*
3	**0.0543**	*1-3-4-5*	35	*1-3-4-5*	35	*TDMA*
4	**0.0550**	*1-5-3-4*	30	*1-5-3-4*	30	*TDMA*
5	**0.0513**	*1-5-4-3*	30	*1-5-4-3*	30	*TDMA*

$\overline{MSE}$ – Valor médio do Erro Médio Quadrático
B.W. – Largura de Banda (*Bandwidth*)
I_1 – Nó Interferência 1; I_2 – Nó de Interferência 2; A – Actuador; S – Sensor; C – Controlador

Tabela 5-11 – Comparação do erro quadrático médio nos cenários 2 e 3/2 para os diferentes protocolos de acesso ao meio

Relativamente ao protocolo *Token Bus*, o impacto do tráfego de interferência é relativamente baixo, podendo ser mais elevado com a utilização de sequências de transmissão de token desfavoráveis. Estas sequências parecem ter uma influência substancial no desempenho do controlo. Um factor que introduz um impacto claro embora pequeno é o número de nós, devido ao aumento do tempo entre duas visitas consecutivas do *token*.

Quando se usa o protocolo *TDMA* podemos verificar que não existe qualquer influência no controlo do facto de se adicionar carga à rede. Neste caso, os factores que condicionam a qualidade do controlo são o número de nós a transmitir na rede, isto é, o ciclo de *TDMA*, os instantes em que cada nó transmite relativamente ao ciclo *TDMA* e ainda a duração dos *time-slots*. Ao alterar o ciclo de transmissão *TDMA*, colocando o controlador a transmitir antes do sensor, verificou-se que o desempenho do controlo melhorou, obtendo mesmo um desempenho idêntico à situação em que existem poucos nós a transmitir. Este facto resultou de uma melhor adaptação entre os instantes de transmissão e as *time-slots* dos nós.

Capítulo VI

Conclusões

Ao longo desta dissertação foram abordadas as características gerais de dois tipos de arquitecturas usadas em sistemas de controlo: por um lado a abordagem centralizada e, por outro, distribuída, caracterizada pela existência de vários nós interligados por uma rede de comunicação.

Com o aumento dos níveis de integração electrónica, torna-se possível construir nós com muito poucos componentes. Assim, devido às suas reduzidas dimensões e baixo custo surge a possibilidade de instalar arquitecturas distribuídas de controlo mesmo dentro de máquinas ou outros sistemas a controlar, dando origem aos sistemas distribuídos e embutidos, e que têm conhecido grande popularidade nos últimos anos.

Um sistema de controlo distribuído genérico inclui pelos menos três funcionalidades distintas normalmente designadas por sensor, controlador e actuador. Contudo, existem várias possibilidades de mapear estas funcionalidades sobre os nós físicos do sistema. As possibilidades abordadas foram: sensor, controlador e actuador num único nó; sensor e controlador num nó diferente do actuador; e sensor, controlador e actuador em diferentes nós. Em configurações distribuídas, estas funcionalidades encontram-se normalmente interligadas por uma rede partilhada, designada por

barramento ou *bus*. Entretanto, a existência de uma rede partilhada possui uma influência determinante no que respeita aos atrasos temporais (latência) na malha de controlo, devido, por exemplo, aos atrasos no acesso ao meio e à carga de comunicação em cada instante. Assim, foram realizadas algumas experiências com o objectivo de compreender melhor o impacto da latência induzida pela rede de comunicação na qualidade do controlo.

Verificou-se que quanto maior o nível de comunicação, i.e., número de mensagens, envolvido na malha de controlo maior é a degradação do respectivo desempenho. Isto acontece por causa dos atrasos devidos à comunicação e também dos atrasos de processamento de tarefas internas a cada nó, sendo todos estes atrasos serializados. Pôde-se comprovar este facto através da análise dos valores do erro médio quadrático obtidos por simulação para os cenários centralizado, distribuído com dois nós e distribuído com três nós. Foi sempre utilizado o mesmo processo, o mesmo controlador e o mesmo protocolo de acesso ao meio (*CSMA/AMP – CAN*), com a taxa de transferência de 125Kbps. Este resultado, esperado, corresponde a afirmar que que quanto maior a distribuição de uma malha de controlo maior o impacto da rede no respectivo desempenho.

Usando ainda como referência o protocolo de acesso ao meio *CSMA/AMP* (*CAN*) adicionaram-se à rede nós de interferência, i.e., que geravam tráfego adicional não relacionado com a malha de controlo. Verificou-se que a degradação do sinal de controlo aumenta significativamente, mesmo para reduzidos níveis de tráfego de interferência. Note-se que o tráfego de interferência segue um padrão estocástico que, embora com probabilidade que pode ser forçada a um valor baixo, permite a ocorrência de *bursts*, i.e., de sequências de transmissões sucessivas.

Finalmente, repetiram-se as experiências de interferência utilizando diferentes protocolos de controlo de acesso ao meio (*MAC Protocols*) mas usando apenas o modelo em que as diversas funcionalidades se encontram dispersas pela rede, com três nós. Usaram-se os protocolos *CSMA/CD* (semelhante a *Ethernet*), *Round Robin* (semelhante a *Token Bus*) e *TDMA* (semelhante a *TTP – Time Triggered Protocol*).

Das comparações entre os vários protocolos ressalta uma diferença fundamental entre os protocolos assíncronos, isto é, do tipo CSMA que transmitem imediatamente sempre que o meio está disponível, e os protocolos com controlo de transmissão (*Token-Bus* e *TDMA*). Enquanto que os primeiros reagem de forma muito eficiente na ausência de tráfego de interferência, já que não induzem nenhum atraso adicional nessas circunstâncias, os segundos sofrem sempre de problemas de sincronização entre os instantes de transmissão dos nós e as janelas de transmissão permitidas pelo

protocolo de acesso ao meio, o que pode gerar alguns atrasos, embora normalmente reduzidos.

Por outro lado, quando se adiciona tráfego à rede, os protocolos do primeiro tipo ressentem-se mais, exibindo atrasos cada vez maiores. Os protocolos do segundo tipo têm limites rígidos para a interferência máxima que cada nó pode causar, de modo que são mais robustos, mostrando uma quase insensibilidade ao tráfego de interferência.

Outro aspecto a ter em conta é a disponibilidade da rede para o tráfego adicional, aspecto este que influencia a eficiência da utilização da largura de banda da rede. Nos protocolos do primeiro tipo (*CSMA*), a rede está normalmente livre e acessível para ser usada por outro tipo de tráfego qualquer. No protocolo *Token-Bus*, a rede não está disponível durante as transmissões do *token*, de modo que convém que os *tokens* sejam o mais curtos possível. Finalmente, com *TDMA* cada nó tem apenas e unicamente o seu *time-slot*, não sendo possível utilizar os outros *time-slots* mesmo quando estejam livres.

Entre os protocolos assíncronos, os que resolvem as colisões apenas por meio de retransmissões, como o *CSMA*/*CD*, apresentam o fenómeno de *thrashing* em que, a partir de certa taxa de ocupação, a capacidade de comunicação efectiva da rede diminui e os atrasos de comunicação aumentam excessivamente. Contudo, garantindo que a ocupação da rede é relativamente baixa, este protocolo também apresenta características aceitáveis. Pelo contrário, os restantes protocolos permitem atingir taxas de ocupação da rede bastante elevadas mantendo um desempenho adequado a aplicações de controlo distribuído.

Em suma, os resultados apresentados nesta dissertação poderão ser um contributo para o projecto de sistemas de controlo distribuídos, ajudando a entender melhor o impacto que a utilização de determinados protocolos de controlo de acesso ao meio têm sobre o desempenho de uma malha de controlo realimentado e, em particular, dos compromissos que estão em jogo.

Numa perspectiva futura seria interessante implementar no simulador *TrueTime* outros protocolos desenvolvidos mais recentemente, tais como por exemplo o *FTT-CAN* (*Flexible Time-Triggered on CAN*), e testar através de simulação o seu comportamento relativamente aos protocolos já estudados.

Bibliografia

Albert, A., (2004). Comparison of Event-Triggered and Time-Triggered Concepts With Regard to Distributed Control Systems. *Embedded World*, Nurnberg, pp 235-252, 17-19 February.

Almeida, L., (1999). Flexibility and Timeliness in Fieldbus-Based Real-Time Systems. *Phd Thesis, University of Aveiro*, Portugal.

Almeida, L., F. Santos, P. Fonseca, J. A. Fonseca, A. Mota (1998). Sistemas de Controlo Distribuído Baseados em CAN: dos Fundamentos à Aplicação.

Almeida, L., (2003). A Word for Operational Flexibility in Distributed Safety-Critical Systems. *Proc. of the IEEE Workshop on Object-Oriented, Dependable and Real-Time Systems (WORDS 2003). Gualajara, Mexico, January 2003*.

Almeida, L., P. Pedreiras, J. A. Fonseca (2002). FTT-CAN: Why and How. *IEEE Transactions on Industrial Electronics*, 49(6), pp 1189-1201, December.

Åström, K. J., e T. Hägglund (1995). PID Controllers: Teory, design and tuning. *Instrument Society of America*. Research Triangle Park. North Carolina.

Babaglou, Ö., R. Drummond (1997). Almost no Cost Clock-Synchronization. *Proc. 17th IEEE International Symposium on Fault-Tolerant Computing*, FCTS-17, Pittsburgh, PA, USA.

Bogenberger, F., B. Müller, and T. Füher et al., (2002), "Protocol overview," *proceedings of the 1st FlexRay International* Workshop, April 16-17, Munich, Germany.

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, K. Årzen (2003). How Does Control Timing Affect Performance? *IEEE Control Systems Magazine*, pp 16-30, June.

CENELEC (1996). European standard EN 50170. *Fieldbus: Vol.1: P-Net; Vol.2: PROFIBUS; Vol.3: WorldFIP*. CENELEC, European Committee for Electrotechnical Standardisation.

DIN (1990). German standards 19245-1 to 19245-3. *PROFIBUS: Process fieldbus*. DIN, Deutsches Institut für Normung.

DS (1990). Danish Standard, DS 21906. P-Net: *Multi-master, multi-net fieldbus for sensor, actuator and controller communications.*

Ferreira, J., P. Pedreiras, L. Almeida, J. A. Fonseca (2002). The FTT-CAN Protocol for Flexibility in Safety-Critical Systems. *IEEE Micro*, pp 46-55.

Fonseca, J. A. (1999). Redes de comunicação em ambientes industriais. Chp. 2, Universidade de Aveiro, Setembro.

Franklin, G., J. Powell, M. Workman (1997). *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 3rd edition.

Hoyme, K., K. Driscoll (1992). SAFEbusTM, *11th AIAA/IEEE Digital Avionics Systems Conference*, Seattle, WA, October.

Henriksson, D., A. Cervin, K. Årzen (2002). TrueTime: Simulation of Control Loops Under Shared Computer Resources. *15th IFAC World Congress on Automatic Control*, Barcelona, Spain, July.

IEC (1998). IEC Standard 61158-3,4: *Fieldbus standard for use in industrial control systems – part 3: Datalink service specification; - part 4: Data link protocol specification*. IEC, Int. Electrotechnical Committee.

ISO/WD11898-4, (2000). Road Vehicles - Controller Area Network (CAN) - Part 4: Time-Triggered Communication, December.

Kim, H. S., (2000). A Systematic Design of a Real-Time Robot Control System Based on a Fieldbus Communication Network. *Proccedings on the 13th CISL Winter Workshop- Sul-Ak Mountain*. February

Kopetz, H., (1993). Should Responsive Systems be Event-Triggered or Time-Triggered? *IEICE transactions on Information and Systems,* E76-D.

Kopetz, H. and G. Grünsteidl (1994). TTP - A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer*, **27(1)**.

Kopetz, H. (1997). Real-Time Systems: Design Principles for Distributed Embedded Applications. *Kluwer Internetional Series in Engeneering and Computer Science*. Kluwer Academis Publishers.

Kopetz, H. (2000). Composability in The Time-Triggered Architecture. *Technische Universität Wien, Austria*

Lönn, H., P. Pettersson (1997). Formal Verification of a TDMA Protocol Start-up Mechanism. *In Proceedings of the Pacific Rim International Symposium on Fault-Tolerant Systems*, pages 235-242. *IEEE Computer Society Press*, 1997.

Luck, R., A. Ray (1990). An Observer-Based Compensation for Distributed Delays. Automatica, 26(5), pp 903-908.

MacGregor, J. (1976). Optimal Choice of the Sampling Interval for Discrete Process Control. *Technometrics*, 18(12), May.

Magalhães, L., B. Silva, H. Lemos, N. Nunes, J. L. Azevedo, B. Cunha, P. Fonseca, L. Almeida (2002). CYCLOP – Robot Autónomo Baseado em CAN com Sistema de Visão. *Revista DETUA*, 3(5), pp 454-460, Janeiro.

Malcolm N. and W. Zhao (1994). The Timed-Token Protocol for Real-Time Communications. *IEEE Computer*, **27(1)**: 35-41.

Malcolm N. and W. Zhao (1995). Hard Real-Time Communication in Multiple-Access Networks. *Journal of Real-Time Systems*, **8(1)**: 35-78.

Martí, P., J. M. Fuertes, G. Fohler, K. Ramamritham (2000). Jitter Compensation for Real-time Control Systems.

Martí, P., J. M. Fuertes, G. Fohler (2000). An Integreted to Real-time Distributed Control Systems Over fieldbus.

Martí, P., J. M. Fuertes, G. Fohler, K. Ramamritham (2002). Improving Quality-of-Control Using Flexible Timing Constraints: Metric and Scheduling Issues. *RT Systems Symposium, 23th IEEE*.

Miskowicz, M., (2001). Adaptive Monitoring the Control Variable in the Event-Triggered Control Networks. *FeT'01*, pp 218-224.

Mota, A., F. Santos, J. A. Fonseca (1999). Sistema de Supervisão e Controlo de Fermentação Alcoólica. *JA99 – XX Jornadas de Automatica do Comité Espanhol do IFAC*, Salamanca, Espanha, Setembro.

Mota, A., J. A. Fonseca, P. Fonseca (2000). Systems Modelling and Identification in CAN based Distributed Control Systems, *Proc. of DCCS 2000, IFAC Workshop on Distributed Computer Control Systems*, Sidney, Australia, December.

Peraldi, M. A., J. D. Decotignie (1995). Combining real-time features of local area networks FIP and CAN, in *Proc. ICC'95 (2nd Int. CAN Conf.)*, pp. 8.11–8.21.

Pleinevaux, P. and J.D. Decotignie (1988). Time Critical Communication Networks: Field Buses. *IEEE Network,* **2(3)**: 55-63.

Rushby, J., (2001). A Comparison of Bus Architectures for Safety-Critical Embedded Systems, *CSL Technical Report, SRI International*, September.

Sanfridson, M. (2000). Timing Problems in Distributed Real-time Computer Control Systems. *Royal Institute of Technology*, KTH, Stockholm, Sweden.

Tanembaum, A. S. (2003). *Computer Networks*. Prentice Hall International Edition, 4th edition.

Thomesse, J.-P (1993). Time and Industrial Local Area Networks. *Proc. of COMPEURO'93*. Paris, France.

Thomesse J.P. (1998). The Fieldbuses. *Annual Reviews in Control*, **22**: 35-45.

Thomesse, J.-P., M. Leon Chavez (1999). Main paradigms as a basis for current fieldbus concepts, in *Proc. FeT'99 (Int. Conf. Fieldbus Technology)*, Magdeburg, Germany, pp. 2–15, September.

TTTech, (1999). TTP/C Protocol, Version 0.5, TTTech Computertechnik GmbH, Vienna, Austria.